cadence®

Liberate™ Characterization Reference Manual

Product Version LIBERATE 21.1
May 2021

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

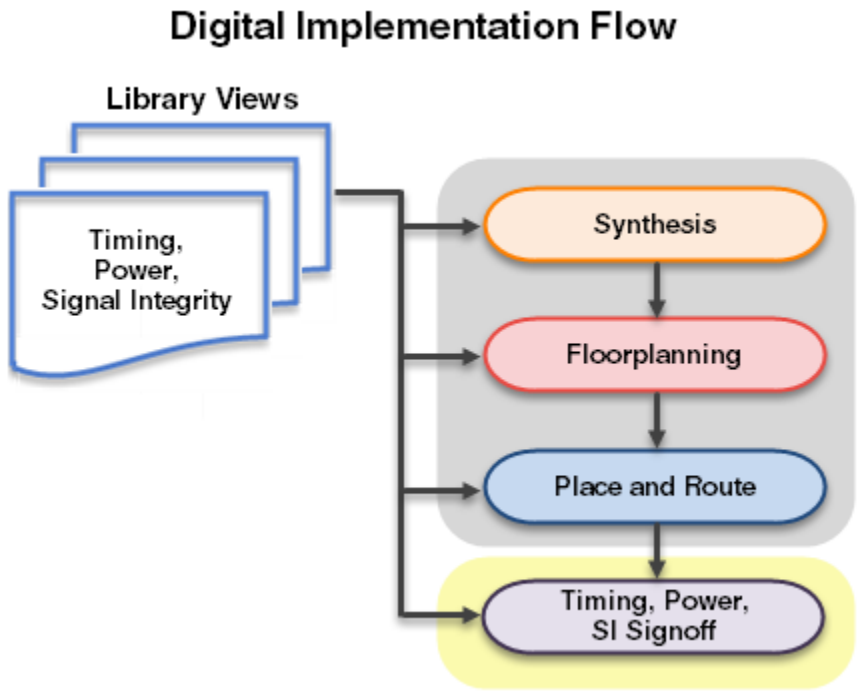
Contents

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.

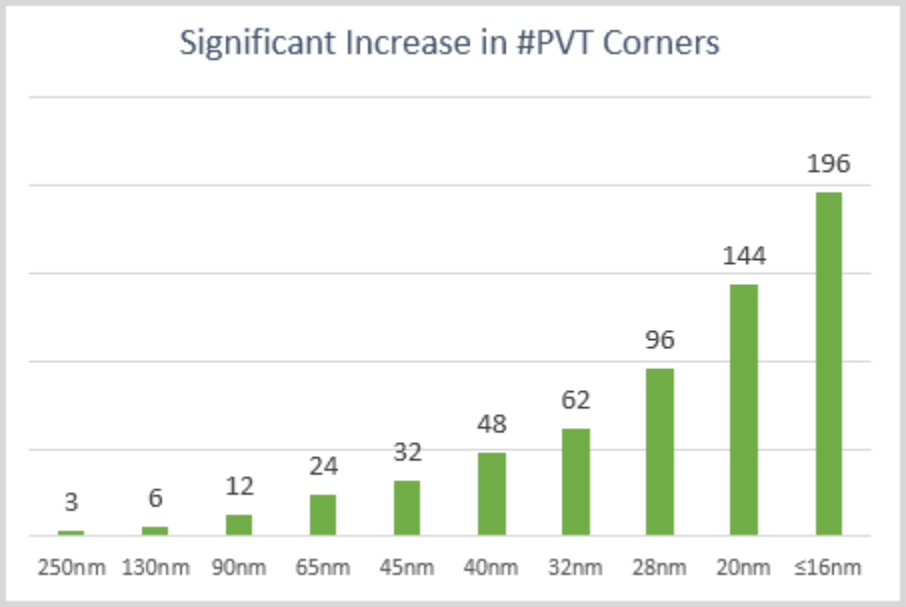


A Growing Problem

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, lower 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 Process, Voltage, Temperature (PVT) corners to accurately model the circuit behavior:

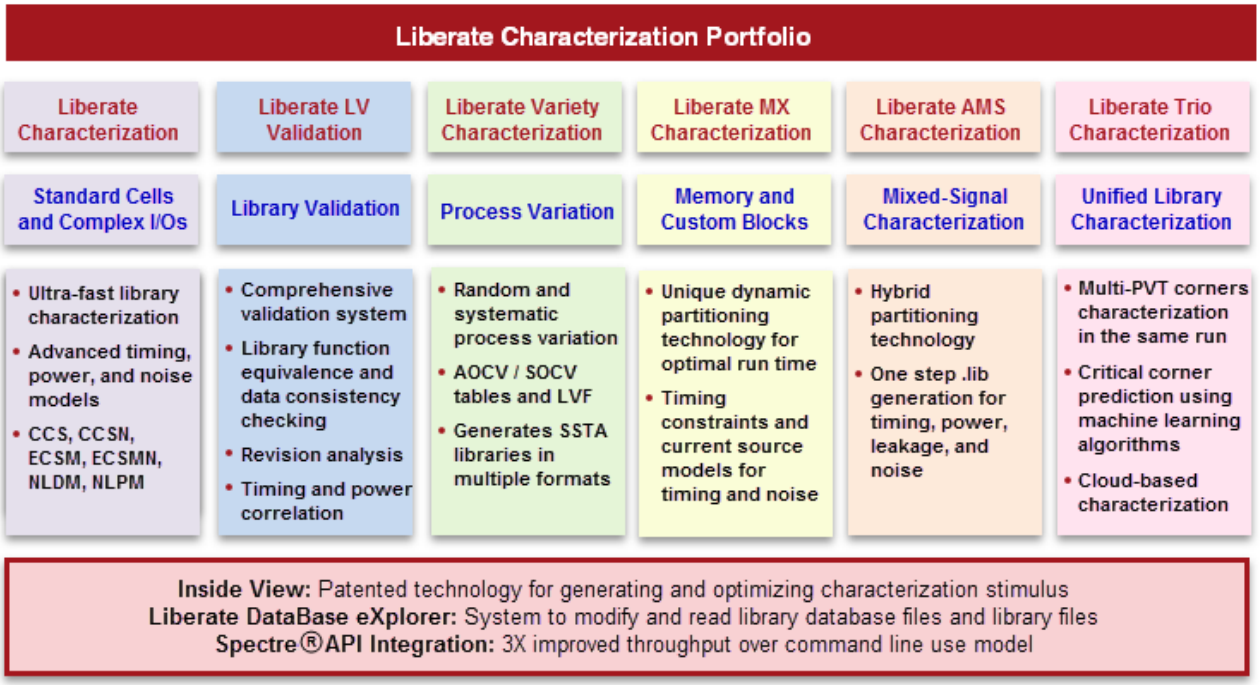


In addition, library views require more advanced models like:

- 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

Liberate Characterization Portfolio

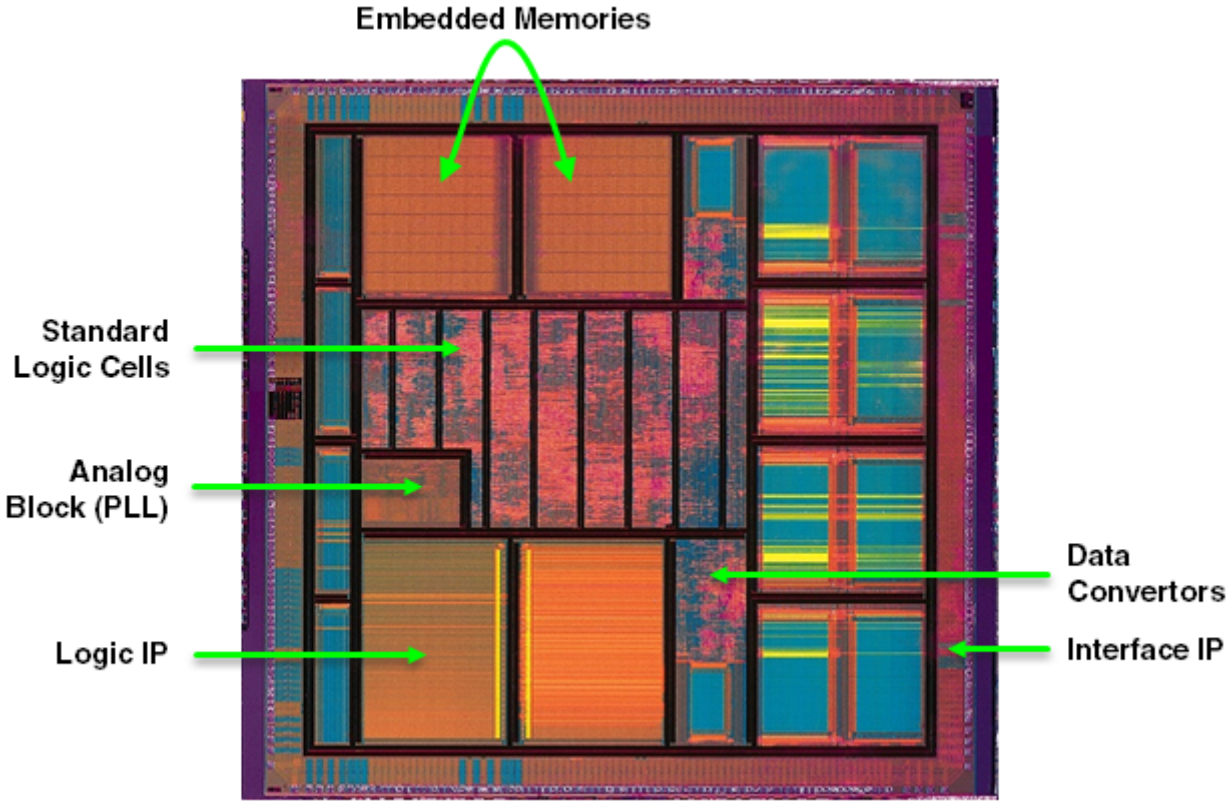
To address all the challenges, Cadence offers Liberate™ Characterization Portfolio that includes the complete set of characterization solutions given below:



The Liberate characterization portfolio intends to provide highly efficient and automated electrical view creation and validation for all IP blocks that including 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, data path, cores, and so on)

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



In addition, the Liberate DataBase eXplorer (Liberate DBX) system of the Liberate characterization portfolio lets you load and manage the contents of library database files (`.ldb`) and library files (`.lib`).

System and Licensing Requirements

Refer to *LIBERATE Software Licensing and Configuration Guide* for information about the different types of software licenses available to use the products of Liberate characterization portfolio. This guide also describes how to configure the licenses for efficient utilization of the available server and client resources.

For detailed information about the system requirements, see Computing Platforms.

About This Manual

The *Liberate Characterization Reference Manual* describes the Cadence® Liberate™ Characterization tool. The manual includes opening chapters that describe what Liberate does and how to get started with the tool. The later chapters discuss the commands and parameters that can be used with Liberate.

Audience Profile

This manual is aimed at developers and designers who want to create electrical views in industry standard formats such as Synopsys Liberty (.lib) format. 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 *Liberate Characterization Portfolio Known Problems and Solutions*.

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

For information about other products in Liberate characterization portfolio, refer to the following manuals:

- *Liberate LV Library Validation Reference Manual* describes the Liberate LV validation tool that provides a collection of capabilities used to validate and verify the data consistency, accuracy, and completeness of cell libraries. This manual also covers information about the Liberate Trio Characterization Suite.
- *Liberate Variety Statistical Characterization Reference Manual* describes Liberate Variety characterization tool that characterizes process variation aware timing models and generates libraries for multiple statistical static timing analyzers (SSTA) without requiring re-characterization for each unique format.
- *Liberate MX Memory Characterization Reference Manual* describes Liberate MX characterization tool that provides library creation capabilities to cover memory cores.

- *Liberate AMS Mixed-Signal Characterization Reference Manual* describes Liberate AMS characterization tool that provides library creation capabilities for Analog Mixed Signal (AMS) macro blocks.
- *Liberate API Reference Manual* describes a Tcl interface that allows access to the Liberate characterized Library DataBase (LDB).
- *Liberate DataBase eXplorer 2.0 Reference Manual* describes the Liberate DBX 2.0 tool that can be used for enhanced post-processing of the data.

Rapid Adoption Kits

Cadence provides Rapid Adoption Kits that demonstrate how to use Liberate characterization portfolio 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`	Indicates text that you must type as presented in the manual. Typically used to denote command, parameter, routine, or argument names that must be typed literally.
`argument`	Indicates text that you must replace with an appropriate argument 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 parameters. 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

What is Liberate?

Liberate is an accurate, highly efficient and easy-to-use library characterizer that creates electrical views (timing, power and signal integrity) in industry standard formats such as Synopsys Liberty (.lib) format. It requires only the foundry device models and the extracted cell netlists (in SPICE format) from which it will create all the required electrical views. By automating the process for generating views, Liberate ensures that the library's functional, timing, power and signal integrity values are both accurate and complete thus avoiding potential chip failures caused by missing or bad library data.

Performance is Key

Given the increase in views, it is paramount that the characterization for nanometer technologies is very efficient. Liberate deploys a number of techniques to improve turnaround time for library creation, including use of a built-in circuit simulator, **AIspice,** that is specially optimized for the simulation of digital circuits. As the creation of a typical standard cell library view often requires over a million simulations, using a simulator which is optimized for characterization can significantly reduce run times. In addition, Liberate uses intelligent techniques to reduce the number of simulations required by eliminating unnecessary vector sequences and deploying smart searching techniques to characterize timing constraints (setup, hold etc.) within sequential cells, which is typically the major bottleneck in the characterization process.

To further improve turnaround times, Liberate supports both multi-threaded and distributed parallel processing. The distribution occurs at a very fine grained level so that the characterization effort is optimally distributed amongst all the available processors.

In addition to using **AIspice**, Liberate supports using external SPICE simulators such as **Spectre®.** Liberate can utilize its internal simulator to reduce the work load of the external

simulator. This maintains consistency with 'golden' circuit simulators while leveraging much of speed advantage of Liberate.

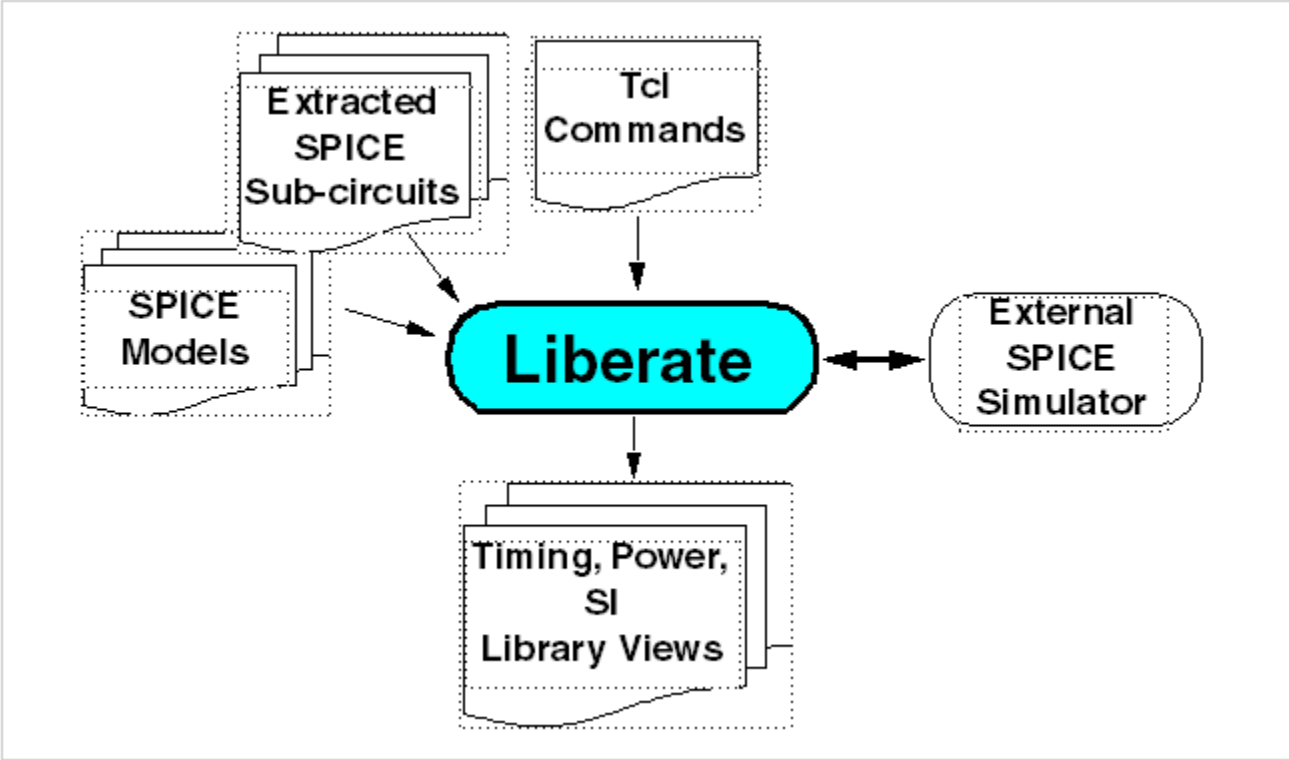


Figure 3: Using Liberate with an external SPICE simulator

Liberate includes the ability to graphically compare libraries. This can be used to verify the results of Liberate against an existing 'golden' library. It also provides early feedback on how foundry model updates will impact a library's electrical characteristics.

2

Getting Started

This chapter describes how to start using Liberate.

Before using Liberate, ensure that it is installed correctly and that all the necessary prerequisite data is available. For information about the prerequisites, see the following manuals:

- *Cadence Installation Guide*
- *Cadence License Manager* manual
- *LIBERATE Software Licensing and Configuration Guide*

Environment Variables

Path to Executable

Set the following environment variables to include Liberate in your executable path:

```
% setenv ALTOSHOME <install_dir>/<liberate_release_name>
% set path=($path $ALTOSHOME/bin)
```

Set the following to include integrated Spectre in your executable path:

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

64-bit Machine Support

Liberate ships with support for 64-bit machines. To use a 64-bit port, set the `CDS_AUTO_64BIT` environment variable to `ALL` before running Liberate, as shown below.

```
% setenv CDS_AUTO_64BIT ALL
```

Alternatively, to launch specific executables in 64-bit mode, you can set the `CDS_AUTO_64BIT` environment variable as shown below:

```
UNIX> setenv CDS_AUTO_64BIT liberate:spectre:variety
```

If you are using Spectre Kernel Interface (SKI), the Liberate and Spectre binaries must be compatible. This means that the same binary style (32- vs 64-bit) and version should be used. To enable 64-bit support in MMSIM:

```
setenv CDS_AUTO_64BIT 1
```

Note: For the version of MMSIM (also known as, Spectre) that is tested for compatibility with Liberate, see the README that ships with the Liberate release. Do not use a version of MMSIM older than the tested compatible version.

Invoking Liberate

Before starting Liberate, you quickly check the version of Liberate by executing the following command on the UNIX prompt:

```
% liberate -v
```

This will print the current version of Liberate and exit.

Liberate utilizes `stdout` and `stderr` for all messages. By default, no log file is created. To invoke Liberate while creating a log file:

```
% liberate my.tcl |& tee my.log
```

It is possible to pass arbitrary strings into Liberate. This is done by added any strings to the Liberate command after the filename. Example:

```
% liberate my.tcl /home/user/liberate/my_run_dir |& tee my.log
```

Then to access the strings in a Liberate run:

```
#puts "Number of Tcl arguments = $argc"
#puts "Tcl Command line = $argv0 $argv"
#for {set i 0} {$i < $argc} {incr i} {
#    set curr_arg [lindex $argv $i]
#    puts "arg $i = $curr_arg"
#}
if {$argc == 0} {
    set run_dir [exec pwd]
} else {
    set run_dir [lindex $argv 0]
}
puts "Set run_dir to $run_dir"
```

Invoking Liberate Help

Help content for the Liberate commands and parameters can be viewed in two ways:

- **In the tool's command prompt**: Suffix the `-help` option to a command name to print a help message listing all the options it supports. Note that if you specify `-help` along with other supported options, only the help content is printed on the tool's command prompt, the other specified options are ignored.

  Note: Some options that get printed may not be officially supported. To view the list of supported options for each command, refer to Chapter 4, "Liberate Commands."

- **In Cadence Help**: On the UNIX prompt, set the path to the LIBERATE installation directory. Then use the `-help` option with the command or parameter name to view the supported and formatted help content in Cadence Help.

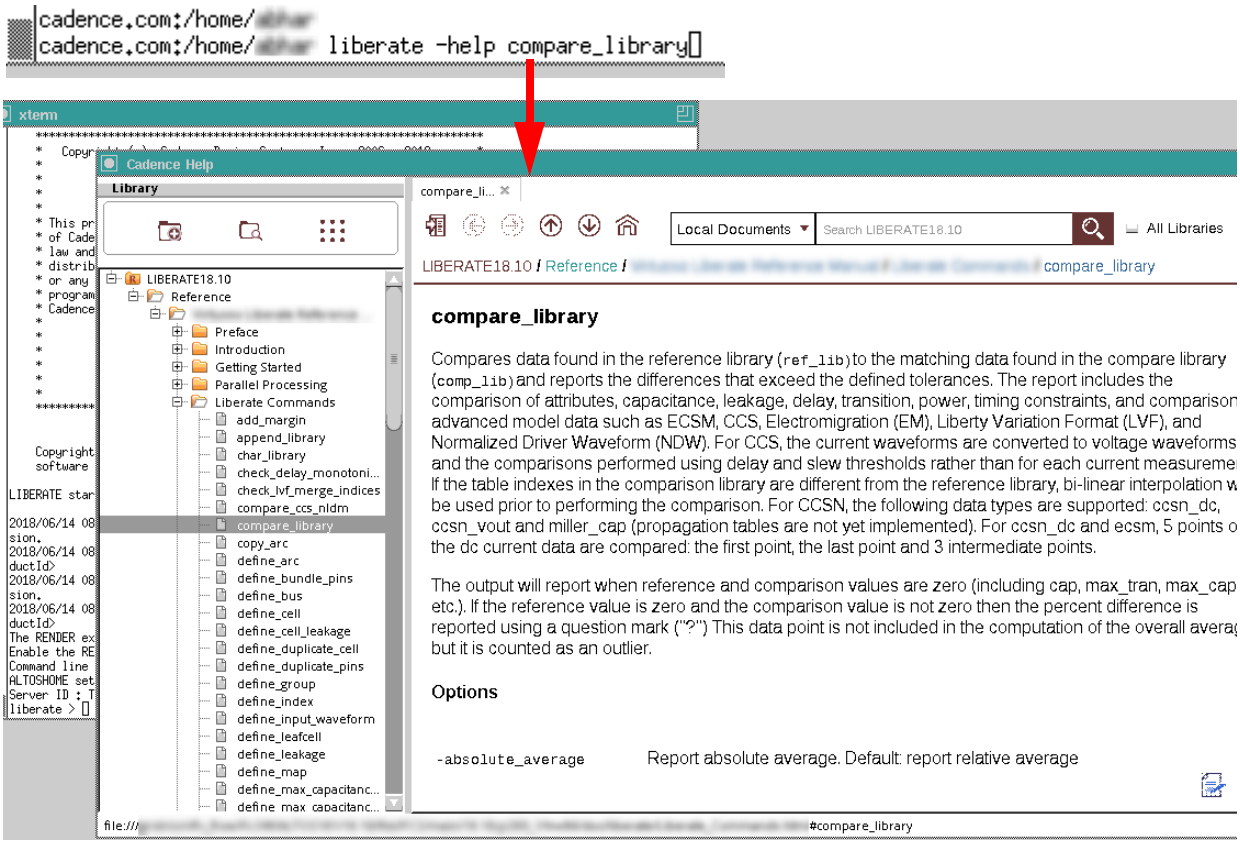
  Use one of the following commands to access help content in Cadence Help:

  - `liberate -help <command_name | parameter_name>`
  - `liberate --help <command_name | parameter_name>`
  - `liberate -h <command_name | parameter_name>`
  - `liberate --h <command_name | parameter_name>`

  Once you are on the tool's command prompt, you can use the help command to view the required help content in Cadence Help. For example:

  ```
  liberate> help compare_library
  ```

This will open the `compare_library` topic in Cadence Help, as shown below.



Caution

When Cadence Help is displayed, the tool command prompt is invoked. Type `exit` on the tool command prompt to return to the UNIX prompt.

Searching Documents for Help Content

You can search for content in Cadence Help using the `-searchdoc` option.

Syntax for using `-searchdoc` in the tool executable command:

```
liberate -searchdoc <search_string>
```

Syntax for using `-searchdoc` in the tool's command prompt:

```
liberate> help -searchdoc <search_string>
```

When either of the above commands is used, the document hierarchy is searched based on the specified search string and the search results are displayed in the Cadence Help window.

This feature serves as an access point through the tool to Cadence Help and allows you to get the search results in one step.

Note: Use double quotes (`" "`) to provide multiple search strings. For example,

```
help -searchdoc "comparison library"
```

System Libraries

When Liberate is shipped, it is enabled with dynamic linked system libraries. To verify that Liberate can run on your system, try to start it as per the procedure described in Invoking Liberate. If Liberate fails to start properly, it may be possible that you have an old system and that there are missing or incorrect system libraries. If this occurs, and you have already checked your environment setup is correct, you can try using static linked binaries by setting the following environment variable. Example:

```
setenv ALTOS_USE_STATIC_BINARIES 1
```

Preparing for Characterization

Three pieces of data are required to run Liberate. These are:

1. Extracted standard cell netlists in SPICE format
2. Foundry device models in SPICE format
3. A Liberate command file in Tcl format

Extracted Cell Netlist

The transistors, diodes, resistors, capacitors and extracted parasitic elements (RCs) that comprise the cell are passed to Liberate in SPICE format. Extracted SPICE netlists can be created directly from the cell layout by device and interconnect parameter extraction tools. Standard SPICE and some Spectre netlist formats are currently 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 Liberate. 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, fast corner, and a slow corner.

To specify the voltage and temperature to use for characterization, use the `set_operating_condition` command. The operating conditions, including the temperature and voltages should be specified before reading in the netlist.

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

To read device models into Liberate, use the `read_spice` Tcl command:

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

Tcl Command File

Liberate 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. Liberate will simulate and measure each cell using each of the specified input slews and loads and will generate the appropriate delay tables, timing checks (setup, hold etc) and power information (switching power, hidden power, state-dependent leakage). Liberate can also generate library information for industry recognized formats such as the Composite Current Source (CCS) model and the Effective Current Source Model (ECSM), among others. For information about the various Tcl commands available for controlling Liberate, see Chapter 4, “Liberate Commands.”

A sample Tcl script for running Liberate is shown below. This script will characterize the cells NAND2x4, NOR2x2 and DFFX1:

```
# Define 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

# Power template for 3 input slews and 3 loads
define_template -type power \
    -index_1 {0.025 0.1 0.25} \
    -index_2 {0.0010 0.015 0.100} \
    power_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

# Read in the SPICE subckts and models
read_spice {models.spi nand2x4.spi nor2x2.spi dffx1.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 \
    -power power_3x3 \
    -constraint constraint_3x3 \
    {NAND2X4 NOR2X2 DFFX1}

# Perform characterization and write out the library
char_library
write_library tt_1p2_25.lib
```

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

```
# Read in an existing library to create templates
read_library existing.lib
write_template liberate_templates
```

The above will create a file called `liberate_templates.tcl`. This file can be used in a subsequent Liberate characterization run. For example:

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

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

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

# Perform characterization and write out the library
char_library
write_library tt_1p2_25.lib
```

Liberate can also be used to recharacterize an existing library to update the models or for a different PVT condition. To do this, use the `-verbose` option with the `write_template` command. This will create a template with additional `define_arc` commands that adhere to the existing library structure. An example is shown below:

```
# Read in an existing library to create verbose templates
read_library existing.lib
write_template -verbose existing_templates
```

The above will create a file called `existing_templates.tcl`. This file can be used in a subsequent Liberate characterization run. For example:

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

# Specify the PVT for this characterization run
set_operating_condition -voltage 1.2 -temp 75

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

# Perform characterization and write out the library
char_library
write_library tt_1p2_75.lib
```

The recharacterization flow shown above honors all arc conditions that pre-exist in a given library. This flow is useful for updating an existing library that Liberate created with PVT (process/temperature/voltage) corner analysis. However, if using a non-Liberate created library, be aware that the input library may not cover all the logical states that Liberate derives from the transistor level description. It is recommended instead to use an existing library to only create the cell and template definitions to be used in a Liberate run. This is to ensure consistent and complete logic-state coverage.

Running Liberate

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

```
% cd $ALTOSHOME/examples/liberate
% liberate char.tcl |& tee char.log
% vi example.lib
```

The example library will contain delay, power and leakage power constructs for a few example cells. To recharacterize this library at a different temperature use:

```
% liberate gen_template.tcl |& tee gen_template.log
% liberate char.tcl |& tee char.log
```

To compare the two libraries use:

```
% echo "compare_library -gui example.cmp.gui \
    -report example.cmp.txt ref.lib example.lib" > comp.tcl
% liberate comp.tcl |& tee comp.log
% vi example.cmp.txt
% lcplot example.cmp.gui
```

It is recommended that library characterization and model generation are distinguished as two separate Liberate runs. This allows for separate optimization of Liberate settings between characterization and model generation.

By default, Liberate keeps all the characterized library data in memory for improved efficiency. However this limits the size of the cell library (~1000 cells) that can be characterized in a single run using 32-bit machines within a reasonable amount of time. The alternative is to use only 64-bit machines or to distribute the work onto multiple hosts. Liberate can automate the distribution using the packet mode. For more information, see the Packet Mode section in Chapter 3, “Parallel Processing.”

The final model generation step (`write_library`, `write_verilog`, and so on.) should then be performed on a 64-bit machine (use `CDS_AUTO_64BIT` environment variable) using the `read_ldb` command. The `read_ldb` command requires a value. This value can be an ldb file or a directory. The `read_ldb` command automatically adjusts and handles the data. There is no option needed (or available) for `read_ldb` to specify if it is loading a file or a directory.

When generating models, the LDB must have all necessary settings stored inside it or the run script that is used to generate the models must load the LDB and provide all necessary settings. To store all commands into the LDB, set the following parameter:

```
set_var  enable_command_history 1
```

Using Spectre with Liberate

Liberate now ships with the version of Spectre that has been qualified for use with the release. See `${ALTOSHOME}/README` for details on the qualified release of Spectre. To use this version of Spectre, make sure your environment is clean and does not include any reference to other versions of MMSIM/Spectre. This includes the `PATH` and the `LD_LIBRARY_PATH` settings.

To use the qualified version of Spectre with Liberate:

➡ Add `$ALTOSHOME/tools.lnx86/spectre/bin` to your environment PATH.

Caution

Do NOT include other versions of MMSIM/Spectre in the `PATH` ***or*** `LD_LIBRARY_PATH` ***settings.***

To use any other specific version of Spectre:

1. Add the `bin` directory to your environment `PATH`.
2. Update your environment settings as needed.

Note: Liberate is designed to allow you to change the version of Spectre used for characterization. It is up to you to ensure that the environment is clean and does not mix different versions.

3

Parallel Processing

This chapter describes how to use Liberate across multiple CPUs.

Cell libraries today typically contain more than 1000 cells with some having as many as 5000 (or more) cells. The cells range from simple inverters to very complex AOI and OAI cells to sequential cells to multibit flop arrays. To characterize these libraries, a server farm is often utilized. In general, the greater the number of CPUs utilized, the faster the run time. Liberate supports concurrent application of both multi-threading and distributed processing.

Multi-Threading

In multi-threading, multiple CPU cores residing on the same physical machine operate on the same memory image. This is the simplest way to use parallel processing with Liberate. The `-thread` argument of the char_library command specifies how many multi-threaded CPUs Liberate can use. The default value of the `-thread` argument is `0`, which allows Liberate to use all of the CPUs on each machine. Liberate does not observe the machine loading and can overload a machine when the number of threads is allowed to default to `0`.

Distributed Processing

Distributed processing occurs when a program distributes the work across many hosts. Liberate partitions the characterization task into a group of related simulations to be performed on each of the available CPUs. Liberate supports the following two forms of distributed processing: Client and Packet.

Using a Queuing System

For starting remote jobs on a client machine, use the rsh_cmd (default `ssh`) parameter to specify the shell command.

Packet Mode

Packet mode pre-processes only those cells that will be characterized on each client machine. This mode is more suitable for a large number of cells. See also Arc Packet Flow.

Arc Packet Flow

When Liberate is called in the Arc Packet flow, a characterization job (the server) is initiated. The server analyzes each cell to sort the cells to be characterized from largest to smallest. Then it sends a request to the queuing system to initialize each client. Set the packet_clients parameter to the total number of clients Liberate can submit. The client does the simulation work. As each client starts, it will notify the server when it is initialized and ready. The server optimally divides the cells to be characterized between the available clients. This means some of the cells will be characterized on a single client while others might be spread over multiple clients. The threshold at which a cell is divided across multiple clients is controlled by the following formula:

```
packet_arcs_per_thread * threads
```

where:

packet_arcs_per_thread is a parameter.

`threads` is the value of the `-thread` argument supplied to the char_library command.

The benefits of the Arc Packet flow are as following:

- Scalability

  Instead of dividing the characterization run into cells, it is now divided into jobs (groups of arcs). Each cell in a library can contain 10 to 200+ arcs. Therefore, we have the choice of running 200 simulations on a single client with 4 CPUs (50 arcs per CPU) or we can run 25 clients with 4 CPUs each (2 arcs per CPU). The simulation wall clock time might be reduced by 25x. However, due to the overhead in preprocessing, there might be a small increase in the CPU time.

- Load balancing

  As each cell is now divided into its component arcs when doing job distribution, the server can balance work better between all available clients.

- Fault tolerance

  The Arc Packet flow has numerous redundancy built-in features, such as following, to handle the potential machine issues:

- All clients communicate regularly with the server and send statistics about load and free memory and so on.
- An unfinished job can be reassigned to a different client.
- A stalled or killed client will be resubmitted to the queue.
- Both the server and the clients will wait when an NFS disk is too slow or busy. The server forces NFS synchronizations for distributed jobs to make sure that files are visible to clients even if the NFS disk system is slow.
- The server regularly monitors the status of all client hosts and prints warning messages if the host is overloaded or about to run out of memory.

Enabling the Arc Packet Flow

The steps to enable the Arc Packet flow are as following:

1. Set up a basic run.

   Run Liberate on a handful of cells without using any batch queue, Arc Packet flow, or SKI. Think of this run as a pipe cleaner run to test the interface to the simulator and the basic characterization settings such as power supplies, templates, netlists, and models.

   When using Spectre, here are some commonly recommended production settings. Your production settings can be different due to changes over time in the recommended settings, and your specific simulation needs.

```
# Set the path to the simulation binary
set_var extsim_cmd "${ALTOSHOME}/tools.lnx86/spectre/bin/spectre"

# Specify the simulator command line arguments
set_var extsim_cmd_option   "+spice"

# Enable the reuse of initial conditions to improve run time
set_var extsim_reuse_ic             3

# The Deck Header is used to include arbitrary Spectre syntax commands.
set_var extsim_deck_header  simulator lang=spectre\nSetOption1 options
        reltol=1e-4\nsimulator lang=spice"

## Simulator Options
set_var extsim_leakage_option "method=gear save=nooutput gmin=1e-15
        redefinedparams=ignore"
set_var extsim_option    "method=gear save=nooutput gmin=1e-15
        redefinedparams=ignore"
```

```
## Tran Append
set_var extsim_tran_append  "lteratio=10"
```

Note: See the README file in the release directory for the compatible version of Spectre. If the Spectre version that you are using is different from the qualified release, a message advising which version is qualified will be printed in the log file.

2. Enable Arc Packet flow.

```
# When using an active driver, enable reuse of the driver waveform
# (see set_driver_cell)
set_driver_waveforms_file driver_waveforms.wave

# For optimal throughput, always use multi-threading.
set THREADS 2

# Specify the maximum number of batch queue submissions (for example, LSF bsub
# commands)
# The total number of Liberate_Client and Simulator licenses required is
# packet_clients * THREADS = 40 for this run
set_var packet_clients   20
set_var packet_mode    arc

# Specify the batch submission command.
# The number of threads should be specified if greater than 1. All threads must
# also run on the same host.
set_var rshcmd "bsub -q qname -R \"span\[hosts=1\]\" -n ${THREADS} -o %B/log
        -e %B/log"
char_library -thread $THREADS -cells $cells
```

3. Analyze the run-time statistics by using the `logs2xlsx` utility.

The `logs2xlsx` utility analyzes the run-time logs from an arc or cell packet characterization run for overall CPU utilization. A report is generated to map the packet run time. This utility takes the following two arguments:

- The path to ldb directory. This can be a full or relative path.
- The path and name of the `.xlsx` file where the results will be saved.

For example:

```
${ALTOSHOME}/bin/logs2xlsx  ${rundir}/my.ldb.gz  ${rundir}/runtime.xlsx
```

The `logs2xlsx` utility automatically detects if the ldb directory contains cell-packet or arc-packet log files, parses them, and saves the data into the named `.xlsx` file. The

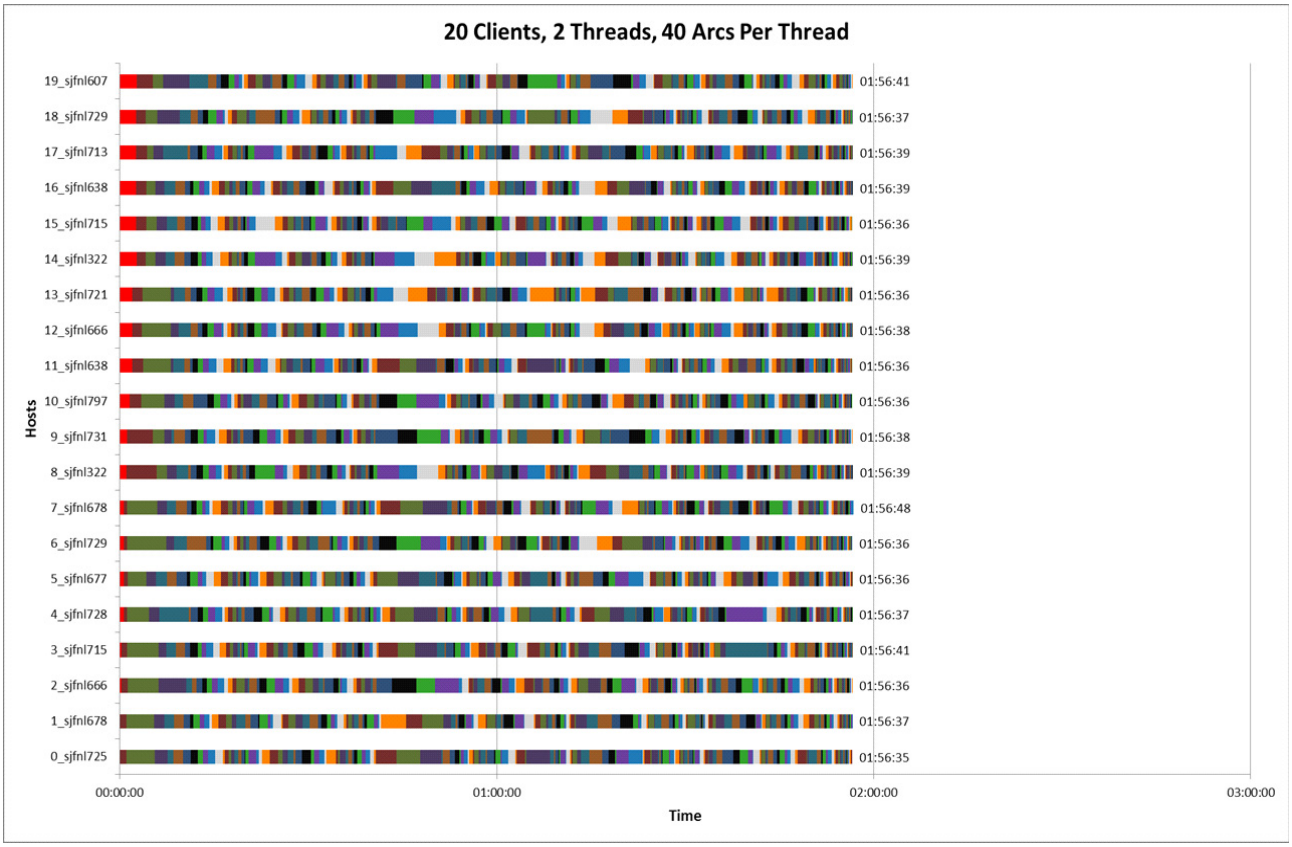
`.xlsx` file can be opened with Microsoft Excel (2007 or later) or OpenOffice Calc on Linux machines. The `.xlsx` file has the following three workbooks (tabs):

- **Summary:** The first section on this tab includes important settings used for arc-packet or cell-packet flow. The second section below it includes a summary for each group of arcs or cell in the arc-packet flow or each cell in the cell-packet flow.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	**Summary for run at:**	../../path/to/run.ldb.gz																		
2	ALTOSHOME	/path/to/LIBERATE14.1/lnx86																		
3	packet_clients	20	Hosts	20																
4	threads	2	Cells	1227																
5	packet_arcs_per_thread	40																		
6	Release	14.1																		
7																				
8	**Cell Name**	**Host**	**Client**	**Task ID**	**Start Date/Time**	**Finish Date/Time**	**Wall Time**	**Error**	**Constraint Arcs**	**Transition Arcs**	**Power Arcs**	**Leakage Arcs**	**Noise Arcs**	**Cap Arcs**	**MPW Arcs**	**Other Arcs**	**Total Arcs**	**Sim CPU Time**	**Total CPU Time**	**Task Latency**
9	CELL_NAME_1	sjfnl713	17	1	09/15/2014 14:24:05	09/15/2014 14:24:21	00:00:16	0	0	10	16	4	0	0	0	0	30	00:00:18	00:00:19	00:00:00
10	CELL_NAME_2	sjfnl728	4	1	09/15/2014 14:24:05	09/15/2014 14:24:24	00:00:19	0	0	10	16	4	0	0	0	0	30	00:00:19	00:00:20	00:00:00
11	CELL_NAME_3	sjfnl677	5	1	09/15/2014 14:24:08	09/15/2014 14:24:26	00:00:19	0	0	10	16	4	0	0	0	0	30	00:00:22	00:00:23	00:00:01
12	CELL_NAME_4	sjfnl678	7	1	09/15/2014 14:24:08	09/15/2014 14:24:32	00:00:24	0	0	10	16	4	0	0	0	0	30	00:00:30	00:00:31	00:00:00
13	CELL_NAME_5	sjfnl666	12	1	09/15/2014 14:24:34	09/15/2014 14:24:48	00:00:16	0	0	4	10	4	0	0	0	0	18	00:00:11	00:00:11	00:00:02
14	CELL_NAME_6	sjfnl638	16	1	09/15/2014 14:24:34	09/15/2014 14:24:47	00:00:15	0	0	4	10	4	0	0	0	0	18	00:00:12	00:00:12	00:00:02
15	CELL_NAME_7	sjfnl638	11	1	09/15/2014 14:24:34	09/15/2014 14:24:46	00:00:12	0	0	4	10	4	0	0	0	0	18	00:00:08	00:00:09	00:00:00
16	CELL_NAME_8	sjfnl797	10	1	09/15/2014 14:24:39	09/15/2014 14:24:51	00:00:12	0	0	4	10	4	0	0	0	0	18	00:00:09	00:00:10	00:00:00
17	CELL_NAME_9	sjfnl715	15	1	09/15/2014 14:24:40	09/15/2014 14:24:51	00:00:11	0	0	4	10	4	0	0	0	0	18	00:00:10	00:00:11	00:00:00
18	CELL_NAME_10	sjfnl322	8	1	09/15/2014 14:24:41	09/15/2014 14:24:55	00:00:14	0	0	4	10	4	0	0	0	0	18	00:00:12	00:00:12	00:00:00
19	CELL_NAME_11	sjfnl721	13	1	09/15/2014 14:24:41	09/15/2014 14:24:53	00:00:12	0	0	4	10	4	0	0	0	0	18	00:00:08	00:00:09	00:00:00
20	CELL_NAME_12	sjfnl729	6	1	09/15/2014 14:24:42	09/15/2014 14:24:54	00:00:12	0	0	4	10	4	0	0	0	0	18	00:00:09	00:00:10	00:00:00
21	CELL_NAME_13	sjfnl729	18	1	09/15/2014 14:24:42	09/15/2014 14:24:55	00:00:13	0	0	4	10	4	0	0	0	0	18	00:00:09	00:00:10	00:00:00
22	CELL_NAME_14	sjfnl728	4	1	09/15/2014 14:24:45	09/15/2014 14:24:58	00:00:14	0	0	4	10	4	0	0	0	0	18	00:00:10	00:00:11	00:00:01
23	CELL_NAME_15	sjfnl797	10	1	09/15/2014 14:19:01	09/15/2014 14:19:25	00:00:25	0	0	14	30	8	0	0	0	0	52	00:00:30	00:00:31	00:00:01
24	CELL_NAME_16	sjfnl728	4	1	09/15/2014 14:19:11	09/15/2014 14:19:35	00:00:27	0	0	14	30	8	0	0	0	0	52	00:00:33	00:00:34	00:00:03
25	CELL_NAME_17	sjfnl729	6	1	09/15/2014 14:19:11	09/15/2014 14:19:40	00:00:34	0	0	14	30	8	0	0	0	0	52	00:00:40	00:00:42	00:00:05
26	CELL_NAME_18	sjfnl322	8	1	09/15/2014 14:19:11	09/15/2014 14:19:43	00:00:32	0	0	14	30	8	0	0	0	0	52	00:00:48	00:00:49	00:00:00
27	CELL_NAME_19	sjfnl322	14	1	09/15/2014 14:22:03	09/15/2014 14:22:20	00:00:17	0	0	6	18	8	0	0	0	0	32	00:00:17	00:00:18	00:00:00
28	CELL_NAME_20	sjfnl678	1	1	09/15/2014 14:22:04	09/15/2014 14:22:21	00:00:18	0	0	6	16	8	0	0	0	0	30	00:00:17	00:00:18	00:00:01

  - **ChartData:** This tab contains the data that is used to create the chart in the next workbook.

- **Chart:** This tab has a bar or column chart depending on the number of machines and total run time. It shows the wall clock time used for each host machine.



Interpreting the logs2xlsx Results

- A run might benefit by changing the packet_arcs_per_thread variable from the default setting.
  - When many hosts are idle at the end, a smaller value might be better.
  - If the colored regions are very small in comparison to the total run time, a larger value might improve the wall clock time.
- The red on the left indicates dead time where the client is waiting to be assigned to a host. If there is significant startup time for each host, there might be significant latency in the queuing system.

File Organization for the Arc Packet Flow

In the Arc Packet flow, the files are organized as described below.

- LDB files
  - The LDB files in the Arc Packet flow are stored in a directory called `altos.<ID>.ldb.gz`. The ID is a Liberate-generated code that is primarily based on the timestamp and the process ID. At the end of the run, use the write_ldb command to rename the temporary directory to a name of your choice.
  - The temporary directory can contain one or more LDB files (one per cell). Each LDB file is named in the format `<CELL>.ldb.gz` where `CELL` is the name of the cell whose characterized data is stored in that particular file.
- Log files

  The client log files in Arc Packet flow capture the stdout and stderr from a particular client and are named in the format `client_<id>.log` where `id` is a number that goes from `0` to `n-1` (`n` is the number of `packet_clients`). As there might also be messages from the queuing system, the LSF log is stored in the LDB in a file called `log.<N>` when requested by the user setting of `-o` or `-e` in the `bsub` command. It should be a copy of `client_<N>.log`, but can also include LSF messages.
- RDB files

  The LDB directory can also contain some RDB data that is used to facilitate distributed characterization of cells. The RDB files get removed after the assembly job completes.

Recovery Flow

If an Arc Packet flow-based run fails to complete, the run can be restarted. To do this, locate the LDB file from the run. Load the LDB file using the read_ldb command before the char_library command and restart the run.

The LDB filename can be controlled using:

```
set_var ldb_filename_prefix MyDB
```

The LDB location can be controlled using:

```
set_var ldb_checkpoint_dir <path>/ldbDir
```

To restart an Arc Packet flow:

```
read_ldb <path>/ldbDir/MyDB.ldb.gz
```

Frequently Asked Questions

- INFO (PRL-36): The time out for a remote job to respond is 3600 seconds.

  Answer: This means that Liberate will wait for 3600 seconds before warning the user that their clients are not getting assigned to a host. If no hosts are being assigned, it might be due to incorrect settings in the rsh_cmd parameter. If a few hosts are getting assigned, the root cause might be an overloaded server farm.

- Performance statistics for each cell show only CPU time. How can one know the wall clock time for each cell?

  Answer: Wall clock time is only relevant for the full run in the Arc Packet flow. Given that all cells "start" at the beginning of the run, the wall clock time for any particular cell is meaningless. You should look at the total wall clock time for all your cells.

set_client Mode (Non-Packet Mode)

The `set_client` mode pre-processes all cells in each client machine. This mode allows Liberate to manage multiple hosts in a distributed processing flow without using a queuing system. However, this mode is not suitable for large libraries because it keeps all characterization data in memory possibly requiring 64-bit software and hosts with a significant amount of memory and swap.

In this mode, the set_client commands specify the names of client host machines to be used. For each machine, a directory in which Liberate can temporarily store data must also be specified.

The rcp_cmd (default `scp`) parameter can be used to specify the command for copying files from the host to the client machines. Before starting a parallel-processing job based on a `set_client` command, ensure that the following commands can be run to access the host machines without requiring any password or passphrase:

- `ssh` or `rsh` from the server to the client (For more information, see the rsh_cmd parameter.)
- `scp` or `rcp` a file from the server to the client (For more information, see the rcp_cmd parameter.)

Following is an example of `set_client` mode:

```
# Specify three client machines to use for job distribution
# Use /tmp/liberate_%N to store intermediate files
set_client -dir /tmp/liberate_%N  LinuxHost1
set_client -dir /tmp/liberate_%N  LinuxHost2
```

```
set_client -dir /tmp/liberate_%N  LinuxHost2
set_var rsh_cmd rsh
set_var rcp_cmd rcp
char_library
```

Bolt Job Distribution System

The Bolt job distribution system is a robust and highly scalable job distribution system that allows high scalability for utilizing thousands of CPUs in parallel on user networks and on cloud infrastructures.

The Bolt job distribution system is supported in Liberate and Liberate Variety only when you are working in Liberate Trio mode. It is also supported in Liberate MX and Liberate AMS.

Architecture

Unlike the traditional job distribution system in Liberate, the Bolt job distribution system is based on a central queuing server called the Bolt server, which runs independently on the Liberate server process. This queuing server manages all the messages between the Liberate server and all the Liberate clients across the network or in the cloud to provide high scalability by maintaining minimal communication traffic with the Liberate server job.

The Liberate server starts the clients across the network and communicates with them through the Bolt server.

Before Liberate can be used with the Bolt job management system, ensure that the Bolt server has already been started. See Starting/Stopping the Bolt Server.

The Bolt server can be manually started on any host. All Liberate servers can then be configured to use that Bolt server by setting the `CDS_BOLT_SERVER` environment variable. See Locating the Bolt Server.

This section is divided into the following two sub-sections:

- Working with Bolt Job Distribution System
- Working with Bolt Servers – For IT Departments

Working with Bolt Job Distribution System

This section explains the working of the Bolt job distribution system from users' perspective.

Enabling Bolt Job Distribution

To enable the Bolt job distribution system, set the following parameter:

```
set_var packet_arc_job_manager "bolt"
```

The following parameters that control the packet arc mode are also supported in the Bolt job distribution system:

- packet_clients
- packet_arcs_per_thread
- packet_client_timeout
- packet_client_timeout_action
- rsh_cmd

When the Bolt job distribution system is in use, you can also use the `nc` command for batch submissions. To enable this support and to ensure that Bolt works properly when the jobs need to be killed, set the packet_rsh_mode parameter to `nc`.

Starting/Stopping the Bolt Server

The Bolt server must be started or stopped manually on a host using the following commands:

- To start the Bolt sever, run:

  ```
  $ALTOSHOME/bin/start_bolt
  ```

- To stop the Bolt sever, run:

  ```
  $ALTOSHOME/bin/stop_bolt
  ```

It is recommended that the Bolt server is started on a dedicated host to keep the server functioning smoothly without any job processing delays.

When the Bolt server needs to be shared among multiple users of Liberate, set the `CDS_BOLT_SERVER` environment variable.

Only one Bolt server can be started per host. Servers can also be formed into server clusters for redundancy. For details, see Working with Redundant Bolt Server Cluster.

When you start the Bolt server using the `start_bolt` command, it checks the maximum open file handles resource on the server host machine. If the number of `(#max_handles-100)` is less than `1000`, a warning is generated to:

- let you know that the Bolt server will be limited to about `(#handles-100)` simultaneous Liberate client connections, and
- suggest increasing the maximum file handles resource if you would like a greater number of connections for this server

An information message is also printed to indicate how many maximum client connections can the server support based on the number of maximum file handles open on the server host.

Locating the Bolt Server

The Bolt server must be started manually on a user-specified host. When the Bolt job distribution system is enabled (packet_arc_job_manager = `bolt`), all Liberate jobs need to know where to access the Bolt server.

To specify the host where the Bolt server is running, set the following environment variable before launching Liberate:

```
setenv CDS_BOLT_SERVER <host_name | IP_address>
```

The value of this environment variable should be the host name or the IP address of the host machine where the Bolt server is running. Liberate will use the provided value as the Bolt server for the current session. If this environment variable is not set and/or the specified server is not accessible, Liberate will exit.

Job Array Support for LSF and Sun Grid Engine

Use of the Bolt job distribution system lets Liberate to scale to a large number of clients. With Bolt, Liberate also provides job array support for LSF and Sun Grid Engine. Job array support is controlled by the bolt_rsh_cmd_use_arrays parameter, which it is enabled by default. When this parameter is enabled, Liberate creates arrays for clients in batches of 1000 and submits an LSF or Sun Grid job for each batch. This reduces the number of system calls and also improves the turnaround time for client submission.

For example, if 1700 clients are requested to be submitted, the jobs will be submitted in two batches:

```
Client 1-1000 submitted. Bsub job id = 2213439
Client 1001-1700 submitted. Bsub job id = 2213440
```

If `bolt_rsh_cmd_use_arrays` is set to `0`, the jobs will be submitted individually.

When support for the `nc` command is enabled, the Bolt job distribution system does not support job arrays.

Controlling Job Arrays: LSF

Example 1: Kill a job array with job ID `555`:

```
bkill 555
```

Example 2: Kill the 10^{th} job in a job array with job ID `555`:

```
bkill "555[10]"
```

Example 3: Kill the jobs from `10` to `15`, `100`, and `369` in a job array with job ID `555`:

```
bkill "555[10-15, 100, 369]"
```

Controlling Job Arrays: Sun Grid Engine

Example 1: Delete a job array with job ID `555`:

```
qdel 555
```

Example 2: Delete the tenth job in a job array with job ID `555`:

```
qdel "555.10"
```

Example 3: Delete the jobs from `10` to `15`, `100`, and `369` in a job array with job ID `555`:

```
qdel "555.10-15, 555.100, 555.369"
```

Qsub Array Support for Sun Grid

In addition to LSF (`bsub`), the Bolt job distribution system provides `qsub` array support for Sun Grid. For example,

```
set_var rsh_cmd "qsub -q queueName -cwd -N jobName -j y -b y"
```

To specify a log file name for `qsub` arrays, writing a `qsub` script is necessary. You can specify the log file directory with the `-o` and `-e` options and `qsub` handles the filenames for array for each individual job. For example, the following command:

```
qsub -o /proj/logdir -e /proj/logdir -t 3 job
```

will start an array of 3 jobs:

```
qsub id = 123456
```

and the log files will be generated in the following format:

```
/proj/logdir/123456.o1
/proj/logdir/123456.o2
/proj/logdir/123456.o3
/proj/logdir/123456.e1
/proj/logdir/123456.e2
/proj/logdir/123456.e3
```

Reporting Bolt Job Distribution Run Status

When the Bolt job distribution system starts, it writes a CSV report file called `run_stats.csv` in the temporary directory of Liberate. The location of this report file is printed in the Liberate server log file. This report file captures a summary of the rolling status while Bolt is running, and puts it into a convenient format for plotting purpose to check the efficiency of the characterization run.

Reviewing the Bolt Log Simulation Files

When you are using the Bolt job distribution system, only the first simulation log file contains the full echo of all parameter settings. The log files generated for the subsequent simulation runs report only the changed parameters. As all simulation log files are saved to the same *<pvt>/<cell>/<log>* directory, you can refer to the first simulation log file in a cell's directory for viewing the related parameter settings. This strategy helps to minimize the disk space usage.

Using Redundant Bolt Servers in Liberate

To take advantage of the redundant servers in a Liberate run that uses the Bolt job distribution system, set the `CDS_BOLT_SERVER` environment variable. The Bolt clusters contain multiple hosts, while Bolt hosts can be standalone or part of a cluster. Therefore, the `CDS_BOLT_SERVER` environment variable supports use of:

- comma (,) to separate a list of hosts that are part of a cluster.
- colon (:) to separate hosts in different clusters.

Below are some examples of setting the `CDS_BOLT_SERVER` environment variable:

- *Example 1: Specify a single Bolt host:*

  ```
  setenv CDS_BOLT_SERVER h1
  ```

- *Example 2: Specify three hosts within a single Bolt cluster:*

```
setenv CDS_BOLT_SERVER h1,h2.h3
```

- *Example 3: Specify two Bolt clusters with three hosts each:*

```
setenv CDS_BOLT_SERVER h1,h2,h3:h4,h5,h6
```

- *Example 4: For backward compatibility, if there are no commas in the list, a single cluster is assumed. Therefore, the following shows three hosts within a single cluster:*

```
setenv CDS_BOLT_SERVER h1:h2:h3
```

In addition, for backward compatibility, there is no way to specify multiple single host clusters. If you must do that, a host can be repeated as shown below:

```
setenv CDS_BOLT_SERVER h1,h1:h2:h3
```

Here, as there is at least one comma in the list, the command is interpreted as specifying three clusters of one host each.

When you specify more than one cluster and a cluster is randomly selected for use, the hosts within the cluster are also rotated randomly to balance the load among all the hosts in the selected cluster. For example,

```
setenv CDS_BOLT_SERVER h1,h2,h3:h4,h5,h6
```

Consider that if cluster `h4,h5,h6` is selected randomly, then the hosts can be shuffled randomly to say, `h5,h4,h6`. So, Liberate Trio server would connect to `h5`. If `h5` becomes unavailable for some reason, `h4` would be tried, followed by `h6`, and then `h5` again before giving up (or if infinite rotation is specified, then retries would continue in this order until connected).

Bolt Client Health Checks

The Bolt job distribution system provides various health checks for Liberate clients. These are enabled by setting the bolt_client_health_checks parameter to `1` (defaults to `0`).

Once the health checks are enabled, the parameters listed below can be used to control various aspects of the health checks:

- bolt_client_cpu_load_threshold
- bolt_client_cpu_memory_min
- bolt_client_cpu_memory_rel
- bolt_client_disk_space_min
- bolt_client_pending_timeout

Tuning Bolt for Advanced Use

To tune the Bolt job distribution system for advanced use:

1. Set the cell priority manually.
   a. Create a CSV file containing a list of cells and the binning priorities.

      The format of the CSV file is `CELL,PRIORITY` on each line, with the `PRIORITY` ranging from `3` (lowest) to `8` (highest). For example:

      ```
      MB16,8
      MB12,7
      MB8,6
      MB6,5
      MB4,4
      MB2,3
      ```

   b. Update the `char.tcl` file to set the bolt_cell_priority_criteria parameter to value, `user_defined`, as shown below:

      ```
      set_var bolt_cell_priority_criteria "user_defined"
      ```

   c. Update `altos_init` or set the `CDS_BOLT_USER_DEFINED_PRIORITIES` environment variable, as shown below:

      ```
      setenv CDS_BOLT_USER_DEFINED_PRIORITIES "/full/path/to/file.csv"
      ```

2. Set the per-arc weighting factors manually using the set_packet_controls command.

Cloud Support

When using Bolt in cloud mode, there are additional licensing check-out options available to allow higher scalability without overwhelming the license server.

Contact Cadence support for details on Cloud settings for Bolt.

Limitations

Bolt job distribution system has the following limitations:

- Job distribution interface support is limited to the local host machine (any number of cores) and distributed hosts using LSF (`bsub`) or Sun Grid (`qsub`).
- The following parameters are not supported in Bolt:
  - `packet_mode`: Bolt only supports arc-based job distribution.

- `packet_arc_optimize_idle_clients`: Clients will always be optimized; clients will automatically shut down when no longer needed or are idle.
- `packet_arc_enable_write_library`
- `packet_arc_notification_list`
- `packet_arc_notification_limit`
- `packet_arc_notification_interval`

Working with Bolt Servers – For IT Departments

This section explains Bolt servers setup, which is usually done by the IT departments.

Bolt Server(s) Hardware Requirements

The following table indicates the Bolt server hardware guidelines. The recommendations are based on the maximum number of clients running simultaneously on a Bolt server or cluster.

Max # of Clients	CPU Cores	Virtual Machine OK?	RAM	Disk Space	Max Open File Handles
500	4	Yes	32 GB	100 MB	1000
1k	4	Yes	32 GB	500 MB	2000
5k	8	No	64 GB	1 GB	10,000
30k	32	No	100 GB	6 GB	60,000
60k	64	No	200 GB	12 GB	120,000

Recommendations:

- Use redundant Bolt server cluster for mission-critical project.
- Run each server on separate physical hosts.

Setting up Bolt Server Host

Perform the following steps to set up a Bolt server host:

1. Change the maximum number of open file handles per host using the root permissions as follows:

   a. Add the following line in the `/etc/sysctl.conf` file:

   ```
   fs.file-max=1208553
   ```

   b. Logout and login again for the changes to take effect.

   c. Run the following to check if the change has applied:

   ```
   # cat /proc/sys/fs/file-max #Should give 1208553
   ```

2. Change the maximum number of open file handles per process using the root permissions as follows:

   d. Add the following line close to end of the `/etc/security/limits.conf` file:

   ```
   * - nofile 819199
   ```

   e. Logout and login again for the changes to take effect.

   f. From bash, check the open file limit per process:

   ```
   Bash>  ulimit –n #Should show 819199
   ```

3. Disable firewall on the server machine and open certain ports that are closed for security.

   The server host(s) should also have certain TCP ports open for Bolt server communication with the Liberate Trio server and clients.

4. Start the Bolt server in the single-server mode with a special user account (or root).

5. As special user, create a working directory where bolt server logs can be written:

   Ensure that the directory is readable:

   ```
   mkdir /mypath/bolt_logs

   chmod 755 /mypath

   chmod 755 /mypath/bolt_logs
   ```

6. Start the Bolt server using this working directory as the log directory:

   ```
   setenv ALTOSHOME <liberate_install_path>

   $ALTOSHOME/bin/start_bolt  /mypath/bolt_logs
   ```

7. Copy the security file into the log area for future use and ensure that it has read permissions. For example:

```
cp ~/.erlang_cookie  /mypath/bolt_logs/

chmod 755 /mypath/bolt_logs/.erlang_cookie
```

Starting Bolt Server Administration

The Bolt server is a RabbitMQ server, which can be administered using the RabbitMQ commands. The following set of commands can be used for Bolt server administration.

Note: This is an advanced usage, typically for IT department.

1. Login to the machine on which the Bolt server is running.
2. Copy the `erlang_cookie` file to your home directory (if you are not running as the same user that started the server):

   ```
   cp /mypath/bolt_logs/.erlang_cookie ~/

   chmod 400 ~/.erlang_cookie
   ```

3. Use the `admin_bolt` command to run the administration commands for the server. For example:

   ```
   setenv ALTOSHOME <liberate_install_path>
   ```

   Run the following command to list all the active queues in the Bolt server:

   ```
   $ALTOSHOME/bin/admin_bolt  rabbitmqctl  list_queues
   ```

Working with Redundant Bolt Server Cluster

For additional fault tolerance, a cluster of Bolt servers can be started on separate hosts such that if one server or host goes down, another server on another host can take over. You can have two or more servers in a Bolt server cluster. Redundant servers must be run on their own hosts.

Note: You cannot run more than one server on a single host machine.

The Bolt job distribution system is built on top of a RabbitMQ 3.7.4 server, which is an open source Message Queuing system based on the AMQP communication protocol. RabbitMQ serves as a vehicle for communication between the Liberate Trio Server and the Liberate clients.

For some common functions, Liberate provides the following Bolt server commands that are based on the RabbitMQ commands:

- General Commands: `start_bolt`, `stop_bolt`, `admin_bolt`

- Cluster Related Commands: `join_bolt`, `status_bolt`

Configuring Bolt RabbitMQ Server

The Bolt RabbitMQ server can be configured using a `.config` file. It can be specified using the following environment variable before starting the server:

`setenv RABBITMQ_CONFIG_FILE <`*`full_path_to_filename`*`>.config`

You can add various options in the configuration file. For example, you can auto rotate the log files based on file size or date:

```
log.file.rotation.date

log.file.rotation.size

log.file.rotation.count
```

For more information on log rotation, see the RabbitMQ website.

Using Bolt Server Management Web Interface

You can also use a Web-based server tool to review server configuration and status. To enable the Web interface, run the following commands before starting the Bolt server:

```
setenv RABBITMQ_ENABLED_PLUGINS_FILE /tmp/enabled_plugins
start_bolt
admin_bolt rabbitmq-plugins enable rabbitmq_management
```

Starting a Redundant Bolt Server Cluster

To start a Bolt server, run the `start_bolt` command as described in Starting/Stopping the Bolt Server. For example, on `host1`, run the following command:

```
host1> start_bolt /root/mq/serverlog
```

Then, start another Bolt server on `host2`:

```
host2> start_bolt /root/mq/serverlog
```

Important

The log directory must be specified in the `start_bolt` commands when using redundant Bolt server clusters. In addition, the log directories for each server MUST be unique. Do not use the same directory name for the log directory for each individual server.

Then, use the following command on `host2` to create a redundant Bolt server cluster and join `host1`:

```
host2> join_bolt host1
```

You can add more hosts using the `start_bolt` and `join_bolt` commands as per your requirement.

This process lets you set up a cluster of redundant Bolt servers.

Checking Status of a Bolt Server Cluster

To check the status of the cluster, use the `status_bolt` command. For example, on `host1`, run the following command:

```
host1> status_bolt
```

The cluster status of node `rabbit@host1` is printed as shown below:

```
[{nodes,[{disc,[rabbit@host2,'rabbit@host1']}]},
    {running_nodes,[rabbit@host2,'rabbit@host1']},
    {cluster_name,<<"rabbit@host1">>},
...]
```

The `nodes` and `running_nodes` options in the status provide a list of all hosts added to the cluster.

Restarting a Bolt Server within an Existing Cluster

If a server within an existing cluster is not working or is taken down for maintenance, you can restart the server to rejoin the existing cluster.

To restart a server on `host2`, use the `start_bolt` command as shown in the example below. The log directory that was used to start the server originally must be specified as a value for the `start_bolt` command.

```
host2> start_bolt /root/mq/serverlog
host2> status_bolt
```

Important

Remember the location of the log directory where the server was started for a particular host, and reuse it to rejoin the cluster. If this is not done, an error is issued regarding a mismatch between the server and the cluster that it is attempting to join.

When the Bolt server is restarted successfully, the status shows that the `running_nodes` are `host1` and `host2`.

Restarting a Single Node in a Bolt Cluster

If a node within a cluster goes down, run the following command to bring it back:

```
node1>  start_bolt <same_log_dir_as_before>
```

Note: The same `log dir` should be used because the cluster information will be present in that directory.

On running the command, the node will join the cluster automatically. There is no need to run the `join_bolt` command.

When node is back up, Liberate will reconnect to this host and continue from where it left if durable queues were enabled using:

```
set_var bolt_use_durable_queues 1
```

To ensure infinite retries, set `bolt_connection_retry_forever=true`. To tell Bolt how long to retry the connection attempts, set `bolt_connection_timeout`.

Restarting All Nodes in a Bolt Cluster

If all the nodes in a cluster goes down, you should bring up nodes in the reverse order that they went down in. Ensure that the same log directory is used. The procedure is same as starting an individual node:

```
node1>  start_bolt <same_log_dir_as_before>
```

For Liberate to continue from where it left, enable durable queues using:

```
set_var bolt_use_durable_queues 1
```

If all the nodes in a cluster goes down together (a rare case), it may not be possible to recover the jobs. In this case, remember the following points:

- When something like a power outage occurs, the logs may get corrupted and it may not be possible to restart the cluster to restore to a previous state.

- Restart each node with a fresh log directory:

  `start_bolt <fresh_log_dir>`

- Use the same new log directory for all nodes in the cluster.

Starting a Cluster Using RabbitMQ Management GUI

You can use a script to start the RabbitMQ management GUI, using which you can start Bolt.

Example

```
<run_bolt.csh>
#!/bin/tcsh -f
setenv RABBITMQ_ENABLED_PLUGINS_FILE /tmp/enabled_plugins
start_bolt $1
sleep 2  ; # wait for full server startup
# enable the RabbitMQ management GUI
admin_bolt rabbitmq-plugins enable rabbitmq_management

# To start Bolt (order is important)
host1> run_bolt.csh <nfspath>/mylogdir
host2> run_bolt.csh <nfspath>/mylogdir
host3> run_bolt.csh <nfspath>/mylogdir
host2> join_bolt host1
host3> join_bolt host1
```

Notes:

- `mylogdir` can be same for all the hosts. A unique directory name will be created under `mylogdir` per host to keep the logs separate.
- When restarting a node, use the same `log dir`.
- The RabbitMQ management GUI is available at any host at:

  `http://hostname:15672/`

Running Arbitrary Administration Commands

For advanced users and IT administration of the Bolt servers, use the `admin_bolt` command that runs a wrapper script for the server-related administration commands. For example, you can run the `rabbitmqctl` administration command as following using the wrapper:

```
host1> admin_bolt rabbitmqctl cluster_status
```

This command dumps the cluster status of the Bolt server running on `host1`.

Clearing Stale Message Queues in Bolt

Sometimes, the message queues in Bolt may not clear up. This might happen in either of the following conditions:

- The Bolt server goes down during a characterization run and the Liberate Trio server is killed, thus preventing the server from clearing up the queues.
- The Liberate Trio servers are killed in a way that interrupts the clean-up part of the exit event loop.

To avoid a queue clutter, the `start_bolt` command adds a *time to live* (TTL) policy of 5-days expiry to all the queues. The TTL policy ensures that the stale queues that have not been used by a server or client for at least 5 consecutive days are automatically expired and deleted. The policy applies to the following types of queues:

- All the existing queues in high-availability or cluster scenarios
- All the durable queues that are loaded when Bolt is started

  Note: The queues that survive a broker restart are classified as durable queues in RabbitMQ. When a node is rebooted, these queues are recovered along with the related messages.
- All newly created queues

This functionality can be overridden by setting the following environment variable before starting the Bolt server using the `start_bolt` command:

```
setenv CDS_BOLT_QUEUE_TIMEOUT <N>
```

where, *N* is an integer number of seconds.

Example

```
CDS_BOLT_QUEUE_TIMEOUT 172800
```

By this setting, all the queues that are inactive for 172800 seconds (2 days * 24 hrs/day * 3600 sec/hr)) will be deleted.

Note: When `start_bolt` is run, the TTL policy for all queues is updated to reflect the new timeout. The time reference for the TTL policy is the start time of Bolt or the last client/server interaction and not the creation time of the queue.

You can also check TTL of a queue in the RabbitMQ GUI. To do so, click the *Queues* tab and select a queue. By default, the queue will show the following information:

```
expires: 432000000" (5 days * 24 hours/day * 3600 seconds/hour * 1000
millisec/sec)
```

Monitoring Bolt Server or Cluster

A Bolt server or cluster can be monitored for health using the `bolt_monitor.py` script, which is shipped with the Liberate installation. This script can be found at the following path:

```
$ALTOSHOME/etc/bolt_monitor.py
```

The functionality of the script is divided into the following two parts:

- Monitor and log: Monitors and logs statistics of each node into a CSV file per node.
- User-specified checks for certain thresholds:
  - Checks a cluster's status and every node within the cluster for its state.
  - Checks monitored values for a user-specified limit.
  - Logs failing events into a user-specified log file (by redirecting the output).

The `bolt_monitor.py` script monitors the following statistics per node in a cluster:

`name`, `running`, `partitions`, `mem_alarm`, `disk_free`, `disk_free_alarm`, `fd_used`, `fd_total`, `mem_used`, `mem_limit`, `proc_used`, `proc_total`, `processors`, `uptime`, `log_files`.

To start the `bolt_monitor.py` script and generate a log file, run the following command on the Unix prompt:

```
% bolt_monitor.py <comma_seperated_hosts_in_cluster> <retry_interval_seconds> >&
<nfs_path>/<cluster_name>.log
```

The `bolt_monitor.py` script outputs error codes that the IT department can monitor to check cluster/node errors and to know when the server needs to be restarted. See Error Codes in bolt_monitor for a list of possible codes. In addition, a log file named `rmqstatsrabbit@<hostname>.csv` file is written to the current directory. This generated log file records the history of the Bolt server as following:

- Bolt server/cluster restart status that reports:
  - If a single host or a host in cluster is down.
  - If an entire cluster is down.

- Notifications that provide information about:
  - Each host (up or down) within a cluster.
  - Network partitions, rabbit alarms (memory, disk, and so on).
  - Memory used, number of file handles, number of processes.
  - Monitor log file per cluster; written to the NFS disk when:
    - A cluster is down or a host within a cluster down.
    - Memory, partitions, or processes are below the user-specified threshold.
    - Any of the RabbitMQ alarms is set.

Customizing Error Thresholds in bolt_monitor

Perform the following steps to customize the error thresholds in the `bolt_monitor.py` script:

1. Make a copy of `$ALTOSHOME/etc/bolt_monitor.py` script and modify the highlighted values of `nodeLimits` in last column of the dictionary.

```
'disk_space'        : [self.ABS_LOW,  1e9,  'GB',  'disk_free',                       5], # GB
'file_descriptors'  : [self.REL_LOW,    1,    '',  'fd_total', 'fd_used',            10], # percent
'processes'         : [self.REL_LOW,    1,    '',  'proc_total', 'proc_used',        10], # percent
'sockets'           : [self.REL_LOW,    1,    '',  'sockets_total', 'sockets_used',  10], # percent
'memory_ram'        : [self.REL_LOW,  1e9,  'GB',  'mem_limit', 'mem_used',          10], # percent
'memory_ram_alarm'  : [self.BOOL,       1,    '',  'mem_alarm',                    True], # string
'total_messages'    : [self.ABS_HIGH,   1,    '',  'queue_totals.messages',     99999999], # messages
'connections'       : [self.ABS_HIGH,   1,    '',  'object_totals.connections', 99999999], # num clients+trio_masters
'channels'          : [self.ABS_HIGH,   1,    '',  'object_totals.channels',    99999999], # num (clients+trio_masters * 3)
'queues'            : [self.ABS_HIGH,   1,    '',  'object_totals.queues',      99999999]  # num (trio_masters*4)
```

   Here,

   - `ABS` is absolute error, `REL` is relative error, and `BOOL` is true or false.
   - If any of these fields fall below (`LOW`) or above (`HIGH`) the user-defined threshold (highlighted values), an error is written out by the script.

2. Change the path to `bolt_monitor.py` script in the copy you made in step 1 above. For this, ensure that the following line is added to the script as its first line:

   ```
   #!$ALTOSHOME/tools.lnx86/python/bin/python
   ```

Where, `$ALTOSHOME` should be replaced with the actual path of your Liberate installation root. For example:

```
#!/home/tools/cadence/liberate192/19.21-s591/tools.lnx86/python/bin/python
```

3. Change the file permissions of the customized version of the `bolt_monitor.py` script. For this, run the following command on the Unix prompt:

```
% chmod 755 bolt_monitor.py
```

4. Run the script from the Unix prompt using the following command

```
% bolt_monitor.py <comma_seperated_hosts_in_cluster> <retry_interval_seconds>
```

If any issues are found during the `bolt_monitor.py` script run, error codes are written to the output. See Error Codes in bolt_monitor for a list of possible codes.

In addition, the `rmqstatsrabbit@<hostname>.csv` file is written to the current directory. The history of the Bolt server is recorded in this `.csv` file that can be imported into Excel and the various statistics can be plotted as needed.

Error Codes in bolt_monitor

The following table lists the error categories and error codes that the `bolt_monitor.py` script might output:

Error Category	Error Code
`Category 0`	The error codes in this category indicate that the cluster might be down: ■ `ERROR_CLUSTER_NODES_NOT_RESPONDING_TO_HTTP_REQUEST` ■ `ERROR_CLUSTER_DOWN`
`Category 1`	The error codes in this category indicate potential network problems or issues with a node that is not responding to a query (this may be temporary). This means that the HTTP API call did not complete successfully as the node might be busy or down: ■ `ERROR_HTTP` ■ `ERROR_HTTP_OTHER` ■ `ERROR_HTTP_QUERY` ■ `ERROR_QUERY`

Error Category	Error Code
Category 2	The following error indicates that a node in the cluster is down: ERROR_NODE_DOWN When a node in the cluster was unable to mirror properly and gets split up as an individual server, the following error is printed: ERROR_NODE_PARTITIONED
Category 3	This error code indicates that a user-defined check (for memory, disk space, and so on) has failed. You configure these in the Python script. ERROR_CHECK_FAILED

Note: In the monitor scripts, a lower error category number indicates a higher severity.

Reporting of Health Incidents

Note: The feature described in this section is available in both the traditional job distribution platform and the Bolt job distribution system.

When a client health incident occurs, that is, if low memory or high CPU load is detected on the remote client machine, the tool will automatically write a report file with the output of the UNIX `top` command. This is useful for checking which user or process is using up the machine's resources.

The report file is named: `$output_dir/ldbs.gz/client_N_HOST_health.rpt`

where, `N` is the client ID and `HOST` is the host that the client is running on.

This file can also be found in the temporary run directory while the client is still running.

Following is the default command that is run when a health incident occurs:

```
"/bin/env HOME=$PWD /usr/bin/top -b -n 1"
```

The output of the `top` command can be customized by placing a file named `.toprc` in the working directory (`$PWD`).

If needed, the command itself can also be overridden by setting the `ALTOS_HEALTH_CMD` environment variable to a desired UNIX command.

This allows flexibility to run other commands or scripts that may produce more information about the health statistics of the client.

To disable writing of health reports, set the environment variable to an empty string as follows:

```
setenv ALTOS_HEALTH_CMD ""
```

Note: With default settings, the `top` command captures information about all users on a particular host and writes it to the report file. If this is not desired, you can disable this feature.

Configuring Shut Down of a Liberate Server

Liberate supports a user-defined script named `altos_exit` that gets executed when Liberate exits. This script is similar to the `altos_init` file that runs before the Liberate binary is launched.

Do not use this script for Bolt servers.

For Liberate to execute the `altos_exit` script, ensure the following:

- The script should be saved in your home directory or the current directory from where Liberate is executed. If a script with this name exists in both the directories, the script residing in the current directory is executed.
- The script must have the Unix execute permissions; otherwise, a Unix error message is displayed.

When the script is executed automatically, an information message is printed to show the script's run status along with the associated exit code.

Enabling Spectre Kernel Interface

The Spectre Kernel Interface (SKI) is used to improve the run time. With SKI, there is a significantly less disk I/O and better license management. Liberate must be run with a compatible version of Spectre. To know which version of Spectre has been qualified for use with the Liberate release that you are using, refer to the README file in the release documents. Do not use a version of Spectre that is older than the qualified version. If an incompatible version of Spectre is in use, a warning message will be printed in the log file and SKI will be disabled automatically.

The* set_client *command cannot be used with SKI.

After checking the Spectre version, run the following steps to enable SKI with Liberate:

1. Run Spectre in standalone mode. A lot of basic issues can be prevented by just getting Liberate working with standalone Spectre.
2. Configure the following environment settings:
   - Both Liberate and Spectre must use the same type of binary (64-bit). Set the following environment variable:

     ```
     setenv CDS_AUTO_64BIT ALL
     ```

   - Use the following Tcl settings:

     ```
     ## Spectre-SKI
     set_var ski_enable                          1
     set_var ski_alter_mode                      3
     set_var ski_clean_mode                      2
     set_var ski_mdlthreshold_exact              1
     set_var ski_power_subtract_output_load_match_extsim     1
     set_var extsim_use_node_name                            1
     set_var extsim_save_passed    none    ; # Do not save the decks with SKI
     as the disk storage requirement is too high
     set ::env(TMPDIR) "/tmp" ; # tmpdir must be set to /tmp SKI, not to $CWD
     ```

     Note: The recommendations can and will change from time to time.

4

Liberate 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.

To access officially supported context-sensitive help information on a command or a parameter from within the tool, follow the procedure covered in the Invoking Liberate Help section.

To review tool-wise support information about each command and parameter available in the Liberate characterization portfolio, see *Liberate Characterization Portfolio Command and Parameter Support Matrix*.

■ interpolate_library	
m…	
■ merge_library	
o…	
■ one_cold	■ one_hot
p…	
■ packet_slave_cells ■ parallelize_tasks	■ printvars
r…	
■ read_ldb ■ read_library ■ read_spice ■ read_training_data	■ read_truth_table ■ read_vdb ■ reset_defaults
s…	
■ select_arc ■ select_index ■ set_aging_criteria ■ set_attribute ■ set_client ■ set_conditional ■ set_constraint ■ set_constraint_criteria ■ set_context ■ set_default_group ■ set_dependent_load ■ set_driver_cell ■ set_driver_waveforms_file	■ set_output_voltage ■ set_packet_controls ■ set_pin_attribute ■ set_pin_capacitance ■ set_pin_delay_threshold ■ set_pin_gnd ■ set_pin_slew_threshold ■ set_pin_vdd ■ set_pvt ■ set_receiver_cap_thresholds ■ set_rsh_cmd ■ set_sim_init_condition ■ set_simultaneous_switch

add_margin

Adds margin (padding) to values in the output library. The margin is added to all cells in the library with two exceptions: power (all types of power) and hidden (input pin power) that can be added to specific cells.

The following steps describe the common usage models:

1. Add margin to the existing timing in the output library. *This is used to add margin to the output library.*
2. Add statistical variation to the existing arcs in the library. *This is often used to add statistical sensitivity as a margin to the existing timing constraints.*
3. Add margin to LVF format data as it is written to the output library.

Options

`-abs <value>`	Specify an absolute margin, that is, the amount of margin to add in standard units. Default: `0.0` (no margin)
`-cells { list }`	List of cells to which the margin should be applied. A wildcard can be used in the following ways: `"*"`, `"xxx*"`, `"*XXX"`, and `"XXX*XX"`.
`-direction < rise | fall | both >`	Specify the data direction to which the margin should be added. Default: "both"
`-ocv`	Use this option to add margin to the `ocv_sigma_*` constructs in an output library (see `write_library -sensitivity_file`). If `-sigma_type` is not specified, the margin is applied to the `ocv_sigma_early` and `ocv_sigma_late` data. To apply a different margin to the `ocv_sigma_early` and `ocv_sigma_late` data, separate Liberate sessions must be used. Use the `-rel` and `-abs` options to specify the margin. When the `-ocv` option is used, the `-type` option supports only `delay`, `setup`, and `hold`.

`-index_1 { list }` Specifies `index_1` points. Default: all points.

All `add_margin -index_1` commands that map to a specific slew/load point(s) get applied.

`-index_2 { list }` Specifies `index_2` points. Default: all points.

All `add_margin -index_2` commands that map to a specific slew/load point(s) get applied.

`-pin { list }` List of pins.

This option accepts the bus syntax, such as `{ A[7:0] }`. It also accepts the asterisk (`*`) wildcard character, as shown below:

```
add_margin -pin addr* #All pins beginning with "addr", such as addr[3]
add_margin -pin * # All pins
```

`-related { list }` List of related pins.

This option accepts the bus syntax, such as `{ 0[7:0] }`. It also accepts the asterisk (`*`) wildcard character.

`-rel <value>` Specify a relative margin. This is a relative ratio amount of margin to add. Default: `0.0` (0%)

If set to a positive number, the resulting library values become larger where larger is defined as more positive. If set to a negative number, the resulting library values are smaller where smaller is defined as more negative.

For example, 0.05 = 5% margin

`-sensitivity_file <filename>`

Specifies the name of the sensitivity file to be loaded. The sensitivity file is created by Liberate Variety (see `write_variation -format "sensitivity"`). The `-sensitivity_file` option can be used only with one of the following `add_margin -type` options: `constraint`, `delay`, `hold`, `non_seq_hold`, `non_seq_setup`, `recovery`, `removal`, `setup`, and `trans`. All `add_margin` commands in a single run must reference the same sensitivity file. If multiple sensitivity files are required, each with different characterized sensitivity values, these files must be merged together into a single file using the `merge_library` command.

Note: Both the `add_margin -sensitivity_file` and `write_library -sensitivity_file` options can be used in the same run, but it is not recommended as special care must be taken to ensure that double counting of sensitivity data does not result.

`-sensitivity_normalize`

Specifies to Liberate that the sensitivity data needs to be normalized to the Liberate LDB corner.

Note: This option can be used only when the `-sensitivity_file` option is used. In addition, the sensitivity file must have been created using the `write_variation` command with the `-format sensitivity_plus_nom` option.

`-sigma_factor <value>`

Specifies a sigma scale factor to apply to the `ocv_sigma_*` data before it is summed to the corresponding nominal data of the specified by `-type`. Only the following `-type` values are supported: `constraint`, `delay`, `hold`, `recovery`, `removal`, `setup`, and `trans`. Default: `1.0`

`-sigma_type { early | late | max | min | avg }`

Specifies the type of ocv_sigma data to be scaled by the -sigma_factor.

`-type { list }`	Specifies the type of data to be modified. Valid types include: `cap`, `constraint`, `delay`, `delay_ccs`, `hidden`, `hold`, `leakage`, `minimum_period`, `mpw`, `nochange`, `non_seq_setup`, `non_seq_hold`, `power`, `recovery`, `removal`, `retain`, `retain_ccs`, `retain_trans`, `setup`, and `trans`. Default: apply margin to all types. If the `-type` option is not specified, the requested margin is applied to all data types. The `constraint` type applies the same margin to ***all*** constraint-related data types. If you use the types `power` and `hidden`, you can also use the `-cells` option to apply the margin to specific cells. However, all other types can only be applied globally. The `delay` type is applied to NLDM and CCS data. The `delay_ccs` type overrides `-type delay` for the CCS offset. **Note:** The types `non_seq_setup` and `non_seq_hold` are supported ***only*** when the `-sensitivity_file` option is also used.
`-when "string"`	Apply the margining to arcs that match the specified state.

This command can be used after **char_library**, read_ldb, and read_library, but it must be used before model generation such as with write_library. Multiple `add_margin` commands can be specified. However, if multiple `add_margin` commands exist with the same type, only the last `add_margin` command is applied. For example:

```
add_margin -cells {DFFX1} -pin {D} -type power -abs 2.0
add_margin -cells {DFFX1} -pin {D} -type power -abs 3.0 <-- only 3 will be added
```

Examples

Example 1:

```
# Add 10% margin to all power data
# Add 50ps absolute margin to all delay data
read_ldb char.ldb
add_margin -type power -rel 0.1
add_margin -type delay -abs 50e-12
write_library margin.lib
```

Example 2:

```
# Read variation sensitivity data, increase the setup sensitivity
# by a factor of 3 and add it to the setup data in the library
read_ldb my.ldb
add_margin -type setup -sigma_factor 3 \
   -sensitivity_file MySensitivity.dat { Cell1 }
write_library -filename my.lib slow
```

Example 3:

```
# Read variation sensitivity data.
# Decrease the non_seq_setup sensitivity to 1/10 of the original value
#   and add it to the non_seq_setup data in the library.
# Decrease the delay sensitivity to 1/2 of the original value and add it
#   to the delay data in the library.
read_ldb char.ldb
add_margin -sensitivity_file "sens.lib" \
   -type "non_seq_setup" -sigma_factor 0.1
add_margin -sensitivity_file "sens.lib" \
   -type "delay" -sigma_factor 0.5
write_library -filename test_margin.lib lib_margin
```

Example 4:

```
# Read process variation sensitivity data.
# Increase the ocv_sigma delay sensitivities by 10%.
# Write out a library with ocv_sigma sensitivity data.
read_ldb char.ldb
add_margin -ocv -rel 0.1 -type delay
write_library -sensitivity_file myLib -filename ocv.lib lib_ocv
```

Example 5

```
add_margin -type {delay} -pin {ZN} -related {I} -cell INV2_12TR35 -abs 2e-9
add_margin -index_2 {0.0004} -type {delay} -pin {ZN} -related {I} -cell
INV2_12TR35 -abs 3e-9
add_margin -index_1 {0.001} -type {delay} -related I -pin ZN -cell INV2_12TR35
-abs 1e-9
```

The point (0.001,0.0004) will map to all three commands. As a result, this point will see a margin of 2+3+1=6ps.

The points (0.001, *) will map to the first and third commands. As a result, these points will see a margin of 2+1=3ps.

The points (*, 0.0004) will map to the first and the second commands. As a result, these points will see a margin of 2+3=5ps.

The other points will map to the first command. As a result, these points will see a margin of 2ps.

add_power_setting

Adds a power setting that has the specified supply condition and a set of supply/grounds settings.

When this command is used, leakage power is characterized and modeled with all settings that are added.

Options

`-cond <condition>`	Specifies the supply condition. Valid value: `VDD*!VSS` or `!VDD+GND`
`-default_only`	Specifies that only a single leakage_power value should be generated for the given power setting. This option is typically used for off-states.
`-levels <levels>`	Specifies a set of supply or ground settings, such as, `{VDD 1.0 VSS 0.0}`.
`<mode>`	Specifies a unique name of the power setting.

The `-cond` and `-levels` options should contain all voltage levels for the library, which can vary depending on the `<mode>`. For modeling a specific cell, the `-cond` and `-levels` options are reduced to contain only the `pg_pins` for that cell.

If `-default_only` is specified, the state-dependent leakage power is not modeled for specified `<mode>`.

If the power name mapping is used (see `set_vdd`/set_gnd `-name_map`), then the `-levels` option should contain the voltage name (from -name_map) instead of the `pg_pin` name. This is because both the `pg_pin` name and the mapped voltage name can be valid voltage names, but with different voltages for a mode.

The `-cond` option should also contain the voltage names, but if it does not, then the mapping to cell-specific supply condition accommodates use of the `pg_pin` name instead.

analyze_advance_aging

Generates the details on how the final aged delay is calculated for a given arc or entry based on encrypted and characterized stress and aging library. This data is useful to debug accuracy while comparing the advance aging library with the single-stress golden aging library.

The following inputs are required:

- Nominal library
- Aging library (could be encrypted) and stress library (encrypted) as a single library or separate libraries
- Stress voltage (VT)
- Arc details

Syntax

```
analyze_advance_aging \
     -nominal_file <file> \
     -aging_file <file> \
     -stress_file <file> \
     -cell <string> \
     -pin <string> \
     [ -pin_dir <R | F> ] \
     -related_pin <string> \
     [ -related_pin_dir <R | F> ] \
     [ -when <string> ] \
     [ -index_1 <int> ] \
     [ -index_2 <int> ] \
     -voltage <float> \
     -temperature <float> \
     -age <float> \
     -duty_cycle <float>
```

Options

`-nominal_file <file>`

Specifies the name of the nominal library file.

`-aging_file <file>` Specifies the name of the aging library file, which could be encrypted or unencrypted.

`-stress_file <file>`

Specifies the name of the encrypted stress file.

`-cell <string>` Specifies the cell related to the arc for which advance aging analysis is needed.

`-pin <string>` Specifies a list of destination pins for the arc (typically, output pins for combinational arcs, input pins for timing constraint, or hidden power arcs).

`-pin_dir <R | F>` (Optional) Specifies the pin direction. Default: Consider both rising and falling transition.

This option can have one of the following values:

`R` Specifies a rising transition.

`F` Specifies a falling transition.

`-related_pin <string>`

Specifies the related pin names.

`-related_pin_dir <R | F>`

(Optional) Specifies the related pin direction. Default: Consider both rising and falling transition.

This option can have one of the following values:

`R` Specifies a rising transition.

`F` Specifies a falling transition.

`-when <string>` (Optional) Defines the logic conditions of the pins of the cell to enable the arc using the Liberty `when` syntax. Default: Match all `when` conditions

`-index_1 <int>` (Optional) Specifies the value to be used as the first index. Default: `1`

`-index_2 <int>` (Optional) Specifies the value to be used as the second index. Default: `1`

`-voltage <float>` Specifies the stress voltage value.

`-temperature <float>`

Specifies the stress temperature value.

`-age <float>` Specifies the calendar age.

`-duty_cycle <float>` Specifies the duty cycle value.

This command must be used before the char_library command is run.

analyze_ccs

Analyzes CCS data. This command can be used to:

- Compare NLDM with CCS, and then output the result in a report
- Compare CCS data with simulation waveform at all thresholds points
- Generate simulation waveform and CCS waveform in a library. These waveforms can be plotted through `gnuplot`.

Syntax

```
analyze_ccs \
     [ -slew_list <list> ] \
     [ -load_list <list> ] \
     [ -pin <string> ] \
     [ -pin_dir <R | F | X> ] \
     [ -related_pin <string> ] \
     [ -related_pin_dir <R | F | X> ] \
     [ -when <string> ] \
     [ -report_dir <string> ] \
     [ -comp_nldm <0 | 1 | 2> ] \
     [ -report_log ] \
     [ -gui ] \
     [ -cell <string> ]
```

Options

`-slew_list <list>`
: Specifies a list of slews to analyze. All slews should be from `index_1` of the lookup table. Default: Analyze all slews

`-load_list <list>`
: Specifies a list of loads to analyze. All loads should be from `index_2` of the lookup table. Default: Analyze all loads

`-pin <string>`
: Specifies a list of pins for CCS analysis. Default: Analyze all pins

`-pin_dir <R | F | X>`
: Specifies the pin direction. Default: Consider both rising and falling transition.

  This option can have one of the following values:

  `R` Specifies a rising transition.

  `F` Specifies a falling transition.

X Specifies both rising and falling transition.

`-related_pin <string>`

Specifies a list of related pins for CCS analysis.

`-related_pin_dir <R | F | X>`

Specifies the related pin direction. Default: Consider both rising and falling transition.

This option can have one of the following values:

R Specifies a rising transition.

F Specifies a falling transition.

X Specifies both rising and falling transition.

`-when <string>` Defines the `when` conditions for CCS analysis.

`-report_dir <string>`

Specifies a directory to store all reports.

`-comp_nldm <0 | 1 | 2>`

Compares CCS with NLDM, checks the final voltage, and generates a report named `compare_ccs_nldm.txt`. Default: `0`

`0`: Do not compare.

`1`: Report the outliers only.

`2`: Report all.

`-report_log` If `-comp_nldm` is specified, setting the `-report` argument generates a warning in the log file if a correlation issue is found while running the `analyze_ccs` command.

`-gui` Generates the waveform curves and a `gnuplot` script file named `gnuplot.cmd`. You can plot the waveform using the following command:

```
gnuplot -p gnuplot.cmd
```

`-cell <string>` Specifies a list of cells for CCR analysis.

This command must be used before the char_library command is run.

Examples

Example 1

```
#analyze CCS, compare ccs with NLDM and generate verbose report
analyze_ccs -comp_nldm 2 -report_log
char_library ...
```

Example 2

```
# analyze CCS for selected arc and selected point
analyze_ccs -pin {Y} -related_pin {A} -when {B} -slew_list {1} -load_list {1} -
report_dir "analyze_ccs”
char_library ...
```

Example 3 (other usages)

```
# generate analyze_ccs directory
analyze_ccs
# only report warning in log, for this ccr request
analyze_ccs -report_log
# generate anaylze_ccs directory; generate compare_ccs_nldm directory(including
outliers)
analyze_ccs -comp_nldm 1
# generate anaylze_ccs directory; generate compare_ccs_nldm directory(including
outliers); report warning in log
analyze_ccs -comp_nldm 1 -report_log
# generate anaylze_ccs directory; generate compare_ccs_nldm directory(including
all entries)
analyze_ccs -comp_nldm 2
# generate anaylze_ccs directory; generate compare_ccs_nldm directory(including
all entries); report warning in log
analyze_ccs -comp_nldm 2 -report_log
```

analyze_ccsp

Writes a report of CCSP and simulation comparison results. In addition, it plots the simulation and CCSP waveforms. The report is saved in the `deck_dir` directory. The plotted waveform is in saved in `decks/ccsp_report/cell/arcID_powername_index_1xindex_2`.

Syntax

```
analyze_ccsp -verbose -format {csv|txt} -report_name <file_name>
```

Options

`-verbose`	Reports all comparison results, writes all simulation waveform data in the LDB, and creates all plot entries. Without the `-verbose` option, the tool reports and plots the outliers.
`-format {csv|txt}`	Specifies the format of the report. The `csv` and `txt` formats are supported. Default: `txt`
`-report_name <file_name>`	Specifies the name of the output comparison file. Default: `comp_ccsp.txt`

This command must be used before the char_library command is run.

append_library

Appends the cells from separate libraries into the output library. The cells should be unique in each library. The action is executed in the following manner:

- Appends the cell-level data "as-is".
- Combines the library-level data (attributes and groups) from multiple libraries including the templates and driver waveforms into a single library header.

Note: The `append_library` command does not merge data from a cell in one library to a cell with the same name in another library. To merge library data, use the merge_library command.

Options

`-allow_conflict`	Allow libraries to be appended even if there are conflicts.
`-filename <filename>`	Output file name. Default: `<library_name>.lib`
`-overwrite`	Overwrite existing .lib file. Default: do not overwrite and create a unique file name.
`-user_data <filename>`	Use this option to add additional attributes and groups to the library-level data of the final library. No attributes are added to the cell-level data.
`{ libraries }`	List of libraries to append.
`<library_name>`	Library name.

Handling Driver Waveform Conflicts

When appending libraries with driver waveforms any conflicts in driver waveforms (`normalized_driver_waveform` groups with the same `driver_waveform_name` in different libraries) are treated as follows:

- **Same index_2** (normalized_voltage) **but different slews** (input_net_transition index_1): If the conflicting normalized driver waveform refers to a different set of slews to the original normalized driver waveform, those slews are merged with the original to form a new normalized waveform group using the original driver_waveform_name name. For example:

If the original normalized driver waveform group with `driver_waveform_name` 'active_driver' has slews (`index_1` entries): **0.1 0.3 0.5**

...and the conflicting waveform group with `driver_waveform_name` 'active_driver' has slews: **0.2 0.4 0.6**

...the output normalized driver waveform group 'active_driver' will include slews: **0.1 0.2 0.3 0.4 0.5 0.6.**

- **Same index_2** (normalized_voltage) **with same slews** (input_net_transition index_1): If the conflicting waveform has some of the same slews as the original waveform, then the waveform values must match with the original waveform within a 1ps tolerance. If all the normalized_driver_waveform values match then no change is made to the original waveform and the conflicting normalized waveform group is not written to the output library.
- **Different values**: If the conflicting waveform has different values (one or more values differ by more that 1ps from the original) then the name of the conflicting waveform is changed to original_waveform_#<num> when <num> represents the number of the library being appended. For example, if the conflict occurs in the 2nd library being appended (3rd in the list passed to append library) then <num> will be 2). All references to the conflicting templates will be renamed to use the new waveform name. A message like this is output:

  ```
  INFO (append_library): Conflicting definition of a driver waveform template
  found in library 'test.lib'. Changing all occurrences of 'ACTIVE-
  WAVEFORM_INV:rise' to 'ACTIVEWAVEFORM_INV:rise_#1' for the cells appended from
  this library.
  ```

  `append_library` permits driver waveform lookup templates to have different index sizes and will output the first definition found, all others ignored.

  The libraries that are not appended under these conditions are:

  - Other template conflicts: If there are other conflicts (a lookup template that uses the same name but refers to a different number of index entries) the library with the conflicting template is not appended. A warning message is output.
  - Voltage map conflicts: `append_library` checks `voltage_map` attributes and omits libraries that have voltage map attributes in conflict with the first definition.

  The `-allow_conflict` option permits libraries with conflicts to be appended. However, resulting library *will have errors* that require fixing before being used by downstream tools.

Example

```
append_library -filename allCells.lib { a_Cells.lib b_Cells.lib } allCells
```

char_library

Performs library characterization. Each cell listed in a `define_cell` command is characterized if the SPICE subcircuit definition for that cell is defined in the netlists passed to the `read_spice` command.

Important

Only one `char_library` command is allowed per Liberate run. This command cannot be run in interactive mode. To characterize a library, add the `char_library` command to a Tcl script file and run the following command on the UNIX prompt to execute this script file: `liberate <tcl_script>.tcl`

In addition to pin capacitance and state-dependent leakage, basic non-linear table lookup models for timing (including effective current source delay models (ECSM), constraints and power are characterized by default.

Note: When writing a library with noise data (ECSMN/CCSN), it is recommended to always include advanced timing (ECSM/CCS) data, that is, always include `-ecsm` (`-ccs`) when using `-ecsmn` (`-ccsn`). This is because the static timers desire a complete set of advanced model data for accurate timing analysis.

Options

`-auto_index`

Instructs Liberate to automatically create the indexes for all constructs (except `si_immunity`) while overriding the values specified in the given templates. The number of entries for each index is taken for the appropriate predefined template. This feature uses the `max_transition` parameter to determine the range of output loads for each cell. To automatically generate `si_immunity` indexes, set the `max_noise_width` parameter.

Important Points to Note

- The `-auto_index` option utilizes the *Inside View* algorithm and ***cannot*** be used with the `-io` option (because the `-io` option disables the *Inside View* algorithm). See Data Table Index Determination for detailed information on determining data table index values.
- If your script includes the `define_index` command, it is applied ***after*** `-auto_index` completes. If `define_index` is successful (correctly specified with cell/pin etc), it will ***override*** the index values determined by `-auto_index`.

If packet mode is being used, the `-auto_index` option requires that the min_transition and min_output_cap parameters are defined. If these are not defined, Liberate generates an error.

Note: In non-Packet Mode, definition of `min_transition` and `min_output_cap` is not needed. However, Liberate outputs a warning that encourages you to set these parameters.

`-auto_max_capacitance`

Instructs Liberate for the explicit computation of the pin-based attribute *max_capacitance* using the same method as the `-auto_index` option. This option incurs the same run-time increase as if `char_library -auto_index` is enabled without modifying the `index_*` values. Only the max_capacitance related attributes are updated. Further, if both *auto_max_capacitance* and `auto_index` are enabled at the same time, then `-auto_index` supersedes.

`-ccs` Instructs Liberate to characterize composite current source (CCS) delay data.

`-ccsn` Instructs Liberate to characterize composite current source noise (CCSN) data.

`-ccsp` Instructs Liberate to characterize composite current source power (CCSP) data. This option is required when advanced power constructs are needed.

`-cells { cell_names }`

Lists the names of cells that need to be characterized. By default, Liberate characterizes all the cells defined by `define_cell` commands**.** When used with the `-exclude` option, Liberate excludes the cells specified in the list from characterization.

`-client_postproc "string"`

Instructs Liberate to execute the given shell command string after a client finishes running in *distributed* mode. For example, this can be used to free up external SPICE licenses as soon as a client finishes rather than waiting until all the clients finish.

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

`-ecsmn`	Enables characterization of the effective current source model noise (ECSMN) data.
`-ecsmp`	Enables characterization of the effective current source model power (ECSMP) data. **Note:** The `-ecsmp` and `-ccsp` options enable the same characterization. Therefore, only one of these two options is required.
`-em`	Instructs Liberate to characterize electromigration (EM) models. This is a standalone characterization that cannot be combined with other data formats such as `-ccs`, `-ccsn`, `-ccsp`, `-ecsm`, `-ecsmn`, and `-ecsmp`. For this feature, you need to use Spectre with APS (see the `-extsim` option of the char_library command and the extsim_cmd_option parameter). **Note:** A netlist must not be flattened by Liberate in the EM flow because the DSPF format files contain information required by the simulator that gets filtered when the netlist is flattened. Because of this, the extsim_flatten_netlist parameter will be overridden to a value of `0`. For more information, see the Electromigration Models section in Chapter 7, "Performing Characterization using Liberate."
`-exclude`	Excludes the cells specified with the `-cells` option from characterization.

`-extsim <simulator_name>`

Instructs Liberate to use the specified external SPICE simulator for characterization instead of Alspice. SKI is the default SPICE simulator. The license for the external simulator must be available. Currently, the following external simulators are supported: Spectre and SKI. If Spectre is specified, the ski_enable parameter can override this setting and enable SKI.

Important

The simulator set using this `-extsim` option must correspond to the setting for the extsim_cmd parameter.

Specifying the `-extsim` option with the `char_library` command creates temporary run directories named `altos.<unique_id>.0`, `altos.<unique_id>.1`, and so on to store the external simulation run-time files based on the thread number such as `0`, `1`, and so on. The `<unique_id>` in the temporary run directory name is a unique ID based on the date, time, and the Liberate process ID.

If distributed processing is requested using the set_client command, these temporary run directories will be created in the directory specified by the `-dir` option of the `set_client` command.

If distributed processing is requested using the packet_log_filename parameter, the `TMPDIR` environment variable and the `tmpdir` parameter determine where the temporary simulation files will be stored. If the specified directory does not exist, Liberate tries to create it using the `mkdir -p` system command.

`-extsim_leakage <simulator_name>`

Instructs Liberate to use the specified external simulator for leakage calculation. This is used for leakage simulations only – all other simulations use the default simulator, Alspice. You can also set the extsim_model_include_leakage parameter to specify a separate set of models for leakage calculations. If this is not set, Liberate uses the same set of models for all simulations.

`-io`

Enables characterization of cells without using the *Inside View* algorithm. This is useful for cells that contain analog or that are too large for digital vector analysis using the *Inside View* algorithm. Using this option disables the automatic arc-determination and vector-generation of Liberate. For I/O cells, each of the arcs and associated logic conditions must be expressed explicitly using the `define_arc` and `define_leakage` commands.

Important

In `io` mode, the `-probe` option is required for all constraint `define_arc` commands, including MPW.

`-lvf`

Enables Liberty Variation Format (LVF) generation from the Liberate run.

Note: This command option is supported only in unified characterization flow of Liberate Trio. For detailed information about this flow, see:

- Chapter 9, "Using the Multi-PVT Characterization Flow of Liberate Trio." of this reference manual, and
- Liberate Trio Licensing section in *LIBERATE Software Licensing and Configuration Guide*

`-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.

`-si`

Instructs Liberate to create signal integrity data. At least a template of type `si_iv_curve` must be pre-defined using the `define_template` command. If a template exists for `si_immunity`, the noise immunity-rejection curves will also be characterized.

`-skip { cin | constraint | delay | hold | hold_only | leakage | mpw | nochange | non_seq_hold | non_seq_setup | power | recovery | removal | setup | setup_only }`

Instructs Liberate to skip characterization of the selected categories. Default: Do not skip any data types

Disables characterization of specific categories of data. A list containing multiple categories is supported. It is recommended that this option should be used only to improve run time while studying the characterization output from Liberate for a specific category. For example, while tuning the setup for constraint characterization, use `-skip { delay power leakage cin hold }` to speed up the run time for constraints.

Skipping a category can have undesirable affects. For example, skipping `delay` and `power` will impact `cin` by reducing the number of vectors used to measure `cin`. We do ***not*** recommend skipping any categories while generating a production library because this can lead to an incomplete library. When the value for the `-skip` option is specified as `setup`, data of the `setup`, `recovery`, and `non_seq_setup` categories is skipped. Specifying the value as `hold`, skips the `hold`, `removal`, and `non_seq_hold` data.

`-skip_list { mycell { list_of_pins } { items_to_skip } <...> }`

Specifies a list of items to skip for a list of cells, or pin-wise for a list of cells.

If more granularity is needed, you can specify a list-of-items to skip on a list-of-pins on a list-of-cells. The skip list can be any item the `-skip` option supports. Examples:

To skip items on cells:

```
char_library -skip_list { mycell { items to skip }
<...> }
```

To skip items on pins on cells:

```
char_library -skip_list { mycell { list of pins } {
items to skip } <...> }
```

The `{ list_of_pins }` supports `*` (asterisk) for a wildcard. Regular expressions such as `"*1"` are not supported at this time. Liberate automatically recognizes if the group after the cell is a list of skip items or a list of pins.

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

Skips characterization for the specified sensitivity type and ensures that it is not modeled in the LDB.
Default: nothing is skipped

Note: This command option is supported only in unified characterization flow of Liberate Trio (see `char_library -lvf`). For detailed information about this flow, see:

- Chapter 9, “Using the Multi-PVT Characterization Flow of Liberate Trio.” of this reference manual, and
- Liberate Trio Licensing section in *LIBERATE Software Licensing and Configuration Guide*

`-thread <number>` Defines the maximum number of threads to use on the current machine. Even if the `-thread` option is not specified Liberate will automatically use multiple threads based on the available CPUs. Running on two or more threads will provide a significant reduction in characterization time.

`-trial`	Generates a dummy database that supports NLDM format library data. This option instructs Liberate to run all of the preprocessing without running the simulations. The database will consist of NLDM format library data with proper structure but dummy data values. All other output formats are disabled including: `ccs`, `ccsn`, `ccsp`, `ecsm`, `ecsmn`, `ecsp`. When combined with a `write_ldb` and `write_library` command, this will result 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. The *Inside View* algorithm of Liberate 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.

The char_library command should never be called after the API is initialized using `ALAPI_init`. All commands that create models call `ALAPI_init`.

Examples

```
# Characterize for CCS and SI
char_library -ccs -si

# Use Spectre for characterization of ECSM and CCSN and CCSP
char_library -ecsm -ccsn -ccsp -extsim Spectre

# Only characterize DFFX1 and INVX1
char_library -cells {INVX1 DFFX1}
```

check_ccs

Converts LIB_CCS to ECSM, converts ECSM to new_CCS, and then compares LIB_CCS with new_CCS. This command also checks the new_CCS for negative time, monotonic, peak, and full_swing issues. The following files are also generated:

- `check_ccs_cmp_report.txt`: It is a comparison report of LIB_CCS and new_CCS.
- `check_ccs_lc_report.txt`: It is a check report of the new_CCS.

Syntax

```
check_ccs -debug
```

Options

`-debug`	Prints all comparison data and output waveform in a log file.

Use the `check_ccs` command after running the `read_library` command.

Example

```
read_library test.lib
check_ccs -debug
```

check_ccsn

Checks the CCB attributes, DC tables (Static CCB, non-monotonic values, bad table, Tempus glitch check), output voltage data and structure, propagated waveform data and structure, arc-based CCSN stages, and both stage-based and referenced CCB format.

Options

`-reltol_miller_cap_ratio <float>`

Specifies the miller capacitance threshold using which the rise and fall ratio is calculated for a cell. Default: `0.1`

`-max_dc_current <float>`

Specifies a current value. Default: `1.0e-06`

The user-defined value is the minimum required value that the maximum DC current should meet.

Setting the `-max_dc_current` option checks whether at least one value in the dc_current table is greater than the user-specified value.

The check passes if at least one value from the dc_current table is greater than the default or user-defined value; otherwise, the check fails and tool reports a warning.

For example:

```
check_ccsn -max_dc_current <value> -lib <file_name>
```

`-reltol_dc_monotonic <float>`

Specifies a monotonicity value. Setting this option checks the dc_current table monotonicity and ROP monotonicity (that is, for monotonically increasing VL, ROP should increase monotonically). Default: `1.0e-02`

Note: When you enable this check, ensure that the `-tempus_glitch_check` option is also set.

A warning is displayed if the dc_current value or glitch ROP is non-monotonic. A warning is also displayed if the glitch ROP does not show monotonically increasing behavior for an increasing VL.

`-reltol_static_dc_table <float>`

Enables the check that analyzes the dc_current table to identify whether it represents the static or non-static behavior correctly.
Default: `1.0e+02`

`-dump {true | false}`

Enables the check that dumps the CCSN structure information from the input library and saves the data to a report file named `check_ccsn.cmp.*`.
Default: `false`

For example:

```
read_library $libname
check_ccsn -dump -lib $libname
```

The generated report contains detailed information about CCB groups, timing groups, and receiver capacitance groups.

The CCB group list has the following details: cell name, pin, `ccsn_first_stage <yes | no>`, `is_needed <true | false>`, `is_inverting <true | false>`, `is_pass_gate <true | false>`, `stage_type <pull_up | pull_down | both>`, `related_ccb_node`, and `when` condition.

The timing group list has the following details: cell name, pin, related_pin, `when` condition, `timing_type <rising_edge | falling_edge | setup_rising/falling | hold_rising/falling | mpw | combinational …>`, and `timing_sense <positive_unate | negative_unate | non_unate>`

The `receiver_capacitance` group has the following details: cell name, pin, and `when` condition.

As the report provides a bird's eye view of all the CCB structures and timing groups, it is useful while debugging and it saves time of going through the whole library.

`-compare {true | false}`

Compares the CCSN structure across multiple CCSN libraries and perform consistency checks on input libraries (same as the `ccsn_compare` script.) Default: `false`

For example:

```
set libs [list reflib cmplib1 cmplib2]
read_library $libs
check_ccsn -compare -lib $libs
```

Two types of reports are generated where one provides a summary report and the other shows the differences in the structure per comparison library. The generated reports contain comparison information like number of CCBs, timing groups, receiver capacitance and so on between the reference and comparison libraries.

`-tempus_glitch_check {true | false}`

Run a CCSN glitch validation check in Liberate. Default: `false`

The `check_ccsn` command reports errors in the first CCB level stage, and LV glitch reported errors in arc level.

The tool constructs a cell model using the miller capacitance, load capacitance, and dc_current table values available from the library. To this cell model, the tool supplies varying input glitches (height, width) known as VL. The range of VL vary from 5% to 95% of VDD.

You can control the number of iterations using the ccsn_tps_glitch_check_iteration parameter.

`-tempus_glitch_check_criteria {<list>}`

Specify the parameters that control the `tempus_glitch_check`, such as shown below:

```
-tempus_glitch_check_criteria {  \
    input_lower_bound 0.2 \
    lower_bound_check_threshold 0.5 \
    input_upper_bound 0.8 \
    upper_bound_check_threshold 0.5 \ }
```

`-lib <lib_name>` Specify a library name.

`-ldb <ldb_name>` Specify an LDB name.

Use the `check_ccsn` command after running the `read_library` command.

Example

Perform check of existing library:

```
read_library $lib?
check_ccsn -lib $lib?
```

Enable check during characterization:

```
…
char_library … $lib
check_ccsn
```

Standalone check of LDB:

```
read_ldb $ldb
check_ccsn -ldb $ldb
```

check_delay_monotonicity

Checks `cell_rise` and `cell_fall` delay data to ensure that all the delay entries are monotonically increasing with respect to output load (index_2). That is, the table is checked for monotonicity for all loads for each slew. The checks are performed as the library is being written out using `write_library`.

Options

`-adjust`	Amount to adjust the delay or transition by when fixing non-monotonic data. Default: `0.001ps`
`-ecsm`	Check ECSM waveform data.
`-exit`	Exit if non-monotonic delay data (by load) is found. This option causes `write_library` to exit if an error is found such that the library is incomplete. 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.
`-fix`	Fix the non-monotonic NLDM data. Use this option to repair any delay and rise/fall transition monotonicity problems (with respect to the output load) by making the non-monotonic table entry equal to the previous entry plus one added to the least significant digit. When non-monotonic data is fixed, it may become out of sync with ECSM and CCS waveform data since only the NLDM data gets fixed. This is especially true when the `-slew` option is used to fix non-monotonic data by slew. **Note:** The `-exit` option overrides the `-fix` option.
`-fix_ccs_delay`	Fix any monotonicity problem caused by output load and/or transition for CCS delay. This option controls whether the CCS delay data should be adjusted. When the library contains both NLDM and CCS timing data, use this option in combination with the `-fix` option. This can help to avoid mismatches between NLDM and CCS timing.
`-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`	Check data monotonicity based on the changing input slew. This means that for a given load, the data corresponding to consecutive all slews are checked. While delay and transition usually slows down with increasing load for a given slew, it is not necessarily correct that there is a correlation between increasing slew and monotonic increasing data for a given load.
`-transition`	Check the rise and fall transitions data.

This command can be used after `char_library`, `read_ldb`, and `read_library`. It should be used before model generation such as with `write_library`.

Example

```
read_ldb my.ldb
check_delay_monotonicity -ecsm -transition -fix
write_library my.lib
```

Warnings and errors will look like:

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

check_lvf_merge_indices

Checks the indexes in the Liberate LDB against the indexes in the sensitivity file for each arc for which both sets of data exist. When merging LVF data into the output library (see write_library `-sensitivity_file`), the indexes must match between the NLDM and LVF data. If the relative and absolute tolerances are exceeded, a message is output. In addition, if the `-action` is set to `error`, Liberate exits without writing out the library.

Options

`-action <"warn" | "error">`

Specifies the action to take when the sensitivity LVF and NLDM indexes do not match within the specified tolerance. Default: `"warn"`

`-slew_abstol <`*`float`*`>`

Specifies the slew (`index_1`) absolute tolerance applied when comparing the index values while merging LVF into the output library. Default: `1e-12` (in seconds)

`-load_abstol <`*`float`*`>`

Specifies the load (`index_2`) absolute tolerance applied when comparing the index values while merging LVF into the output library. Default: `1e-15` (in farads)

`-slew_reltol <`*`float`*`>`

Specifies the slew (`index_1`) relative tolerance applied when comparing the index values while merging LVF into the output library. Default: `0.01` (1%)

`-load_reltol <`*`float`*`>`

Specifies the load (`index_2`) relative tolerance applied when comparing the index values while merging LVF into the output library. Default: `0.01` (1%)

This command must be used before the `write_library` command is run.

compare_ccs_nldm

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

Options

`-absolute_average` Report absolute average. Default: *report relative average*

`-abstol <value>` Sets the absolute tolerance limit for the comparison between CCS and NDLM errors. Default: `0.002 × time_unit` (typically `2ns`)

Note: The `-abstol` and `-reltol` options define absolute and relative tolerance limits for each comparison. Any comparison that exceeds both these tolerances is considered an outlier and will be reported.

`-cells {cell_names}`

Specifies a list of cells to compare. Default: *all cells*

Note: This option supports the use of a wildcard. If the `-exclude` option is used, the list of cells will be excluded from the comparison.

`-exclude` Exclude the list of cells from the comparison.

`-format < txt | xls | htm >`

Specifies the format for the output report, that is, text, Microsoft Excel®, or HTML. Default: `txt`

The default (`txt`) is a standard textual format. The `xls` value can be used to get an output format that is more suitable for import into Microsoft Excel. The `htm` value can be used to request an HTML output format. The default directory name is "`./html`" and can be changed using the `-group` option. A one page comparison will be generated for each cell group. Open the file `index.html` in a web browser to view the report.

`-group <dirname>`	Specifies the directory name to store cell comparisons for each cell group. Default: *all cells in a single report* The `-group` option 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 *footprint* attribute. The comparison report for each group is stored in the file *<dir_name>/<group_name>.cmp.txt*
`-gui <filename>`	Defines the name of an intermediate file that can be used for graphical comparisons of data with the `lcplot` utility.
`-lcplot`	Starts the `lcplot` utility for viewing the comparison results graphically. When using this option, the `-gui` option is not required because a comparison data file called *<library_lib>.gui* will be created automatically.
`-nworst <number>`	List the number of top *nworst* cells with outliers per data type. Default: `5` This option defines the number of failing cells to report for each data type in the report summary. For the top *nworst* cells, the worst absolute and relative outlier is reported.
`-percent_max_diff`	Report the percent error of the max difference. Default: *report the maximum percent difference*
`-reltol <value>`	Percentage tolerance for the limit for the comparison between CCS and NDLM errors. Default: `0.02` **Note:** The `-abstol` and `-reltol` options define absolute and relative tolerance limits for each comparison. Any comparison that exceeds both these tolerances is considered an outlier and will be reported.
`-report <filename>`	Output comparison file name. Default: *<library_name>.cmp.txt* An overall comparison summary is also written to the standard output.
`-verbose`	Report comparison of all data in the library, regardless of tolerances. The generated report shows every comparison including those that did not exceed a tolerance.
`<library_name>`	Library name.

This command should be run by itself in a separate Liberate run.

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 data found in the reference library (`ref_lib`) to 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 indexes 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, 5 points of the `dc` current data are compared: the first point, the last point and 3 intermediate points.

The output will report when reference and comparison values are zero (including `cap`, `max_tran`, `max_cap`, and so on). If the reference value is zero and the comparison value is not zero, the percent difference is reported using a question mark ("?"). This data point is not included in the computation of the overall average but it is counted as an outlier.

Options

`-absolute_average` Report absolute average. Default: *report relative average*

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

Specifies the absolute difference tolerance. The `abstol` value should be given in standard units and instead of library units, such as, `"delay 5e-12"` to set the `abstol` for delay to `5ps`.

Default: `1e-3 × data_type_unit` (`0.001` times the default unit for each data type)

For example, if the `time_unit` is in `nS`, the `abstol` for delay will default to `0.001nS` or `1ps`.

Note: The `-abstol` and `-reltol` options define absolute and relative tolerance limits for each comparison. These options accept a single value or a paired list of type and value. Any comparison that exceeds both these tolerances is considered an outlier and will be reported. 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 has only a single value, the type for that value is assumed to be *all*.

`-cells {cell_names}` Specifies a list of cells to compare. Default: *all cells*

Note: This option supports the use of a wildcard. If the `-exclude` option is used with this command, the specified list of cells will be excluded from the comparison.

`-cellmap {list}` Specifies a list of pairs of `ref_lib` and `comp_lib` cells to compare. Default: Compare all matching cell names.

The new option is used to control how `compare_library` chooses which cells to compare. The rules for priorities pertaining to cell mapping are as follows.

- If both `-cells` and `-cellmap` options are specified, then each cell in the `-cells` list is compared to the `cellmap` list. If a cell in the `-cells` list maps to a valid pair in the `cellmap` then the mapped reference cell and comparison cell is compared. One-to-many and many-to-one mapping is allowed and comparison is done for all valid combinations. If a cell in the `-cells` list is not present in the `cellmap`, then the comparison is done for the same cell in both libraries.
- If only `-cellmap` is provided but not `-cells`, then each valid cell pair from the cellmap is compared.
- If only `-cells` is provided but not `-cellmap`, then all cells in the `-cells` list that exist in both libraries are compared.
- If neither `-cells` and `-cellmap` are provided, then all the cells that are present in both the reference and comparison libraries are compared.

`-comp_adjust_tristate_load <value>`

Controls if pin capacitance should be added to the load indexes on tristate pins and adjusts the tristate load of comparison library before comparing. Default: `-1`

The accepted values are:

`-1`	Follow setting of the adjust_tristate_load parameter.
`0`	Do not adjust the load indexes.
`1`	Add the `rise_capacitance` and `fall_capacitance` of the tristate pin to the associated load indexes.
`2`	The same as `1` except that the pin attribute `capacitance` will be added to the load indexes.
`21`	The same as `1` except that only timing arc loads will be adjusted (not power arc loads.)
`22`	The same as `2` except that only timing arc loads will be adjusted (not power arc loads.)

`-constraint_report_style < all | separate >`

Specifies the constraint reporting style.
Default: `all` (combines all constraint types, such as, setup and hold, into a single constraint data type.)

`-exact_match` Only compare arcs with identical logic ('*when*') conditions.

`-exclude` Exclude the list of cells from the comparison.

`-format < txt | xls | htm >`

Specifies the format for the output report, that is, text, Microsoft Excel®, or HTML. Default: `txt`

The default (`txt`) is a standard textual format. The `xls` value can be used to get an output format that is more suitable for import into Microsoft Excel. The `htm` value can be used to request an HTML output format. The default directory name is "`./html`" and can be changed using the `-group` option. A one page comparison will be generated for each cell group. Open the file `index.html` in a web browser to view the report.

`-group <dirname>` Specifies the directory name to store cell comparisons for each cell group. Default: *all cells in a single report*

The `-group` option 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 *footprint* attribute. The comparison report for each group is stored in the file *<dir_name>/<group_name>.cmp.txt*

`-gui <filename>` Defines the name of an intermediate file that can be used for graphical comparisons of data with the `lcplot` utility.

`-index1_range <range>`

index1 range to use for comparison. Default: *use all indexes*

`-index2_range <range>`

index2 range to use for comparison. Default: *use all indexes*

The `-index1_range` and `-index2_range` options can be used to limit the comparison to a range of *index1* or *index2* values. The range is either two values separated by a "`-`" (for example, "`1-3`" to compare the first three indexes) or a single value (For example, "`2`" to compare the second index only).

`-lcplot` Starts the `lcplot` utility for viewing the comparison results graphically. When using this option, the `-gui` option is not required because a comparison data file called *<comp_lib>.gui* will be created automatically.

`-lib <abs | rel>` Request a Liberty formatted report with absolute or relative data differences.

The `-lib` option can be used to request 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 will be named *<comp.lib>_<abs | rel>.cmp*.

`-merge_only`

Merges the cell-level reports. This option lets you parallelize `compare_library` at cell level on different machines. After the parallel runs are complete, use the `-merge_only` option to compare the library-level attributes and combine the cell-level reports.

Syntax:

```
compare_library -merge_only -cells <cell_list>  -report
$report_dir/my_report.txt
```

Here, `-cells <cell_list>` is supported, but it is optional. It is possible to merge only a subset of cell-level reports using the `-cells` option with `-merge_only`.

Note: This option should be used only when cell-level `compare_library` reports are already available. Cell-level reports are created if `compare_library` is run for a single cell at a time using the `-cells` option.

Following is an example based on parallel tasks:

```
#############
## compare_top.tcl
#############
## parallelize compare_library at cell level
      source ${CELLSFILE}
      foreach cell $cells {
         create_task -script $rundir/compare_bot.tcl
-args [list $cell $REPORTNAME $REFLIB $CMPLIB] -logdir
$rundir/new_data/logs/compare_library/$cell
      }
      parallelize_tasks -workdir $rundir/new_data


## Merge
      set exec_compare_library_merge "compare_library \
                -cells $cell\
                -merge_only\
                -report ${REPORTNAME}.cmp.txt \
                $REFLIB \
                $COMPARELIB"
       puts "${exec_compare_library_merge}"
       eval $exec_compare_library_merge
```

```
#############
## compare_bot.tcl
#############
set cell [lindex $argv 0]
set REPORTNAME [lindex $argv 1]
set REFLIB [lindex $argv 2]
set COMPARELIB [lindex $argv 3]

set exec_compare_library "compare_library \
          -cells $cell\
          -report ${REPORTNAME}.cmp.txt \
          $REFLIB \
          $COMPARELIB"

puts "SCM ${exec_compare_library}"
eval $exec_compare_library
```

`-multiple_matches` Report comparison of all arcs that have functional overlap with a reference arc

`-nldm_only` Requests comparison of only the NLDM data. 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 mismatched numbers of indexes. No interpolation between index points will be performed. By default, if the number of index values is different, the comparison values will be interpolated.

`-nworst <number>` List the number of top *nworst* cells with outliers per data type. Default: `5`

This option defines the number of failing cells to report for each data type in the report summary. For the top *nworst* cells, the worst absolute and relative outlier is reported.

`-ocv_const_report_style < all | separate >`

Specifies how the following OCV data types are reported in the summary file:

- `ocv_const`
- `ocv_const_mean_shift`
- `ocv_const_stddev`
- `ocv_const_skewness`

`all` (Default) Prints a combined entry for each data type for the specified constraint types, such as setup and hold.

Following is an example of summary of outliers:

```
ocv_const has 10 outliers
```

Here, `10` is the sum of outliers of all the specified constraint types.

`separate` Prints separate entry for each data type for all the specified constraint types.

For example:

```
ocv_const(setup) has 5 outliers,
ocv_const(hold) has 5 outliers.
```

`-ocv_include_nominal`

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

The LVF (Liberty Variation Format) 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.

`-ocv_over_nominal` Changes the algorithm used to compute the relative difference for OCV data types to divide the OCV difference by the maximum of the `nominal_delay` or `stddev` from the reference library.

Note: This option is recommended if the `nominal_delay` is small or negative because it can cause very large percent errors.

The changed algorithm divides the OCV difference (`ref_ocv - comp_ocv`) by the maximum of the `ref_nominal_delay` and the `ref_stddev` when computing the relative difference for OCV data types.

`-ocv_sigma_factor <` *`factor | type_factor_pairs`* `>`

Multiplies the OCV data by the given factor before data comparison. The value supports a single value or a paired list of `ocv_type` values. The supported types are: `ocv_delay`, `ocv_trans`, and `ocv_const`.

Examples: `"3"` or `"ocv_delay 3.0 ocv_trans 1.5 ocv_const 2.5"`

Default: `1.0` for all OCV data types

`-padding` Pad delay, transitions and constraints by ½ input slew. Pad power by an additional $\frac{1}{2}CV^2$.

The `-padding` option can be useful when comparing very small or even negative delay values. The reference and comparison delay, transition, and constraint data is padded by a ½ input slew before comparison. In addition, the power values are now padded by an additional $\frac{1}{2}CV^2$ (where `C`=output capacitance, `V`=Vdd for that pin) to the power numbers before performing the comparison. This will not apply to hidden power because the output is not toggling.

`-padding_index` Specify which slew index to use when adding padding. Use one of the following values to specify this: `same`, `mid`, and `end`. The default is to use the `same` slew index as for delay.

`-percent_max_diff` Report the percent error of the max difference. Default: *report max percent difference*

`-ref_adjust_tristate_load <value>`

Controls if pin capacitance should be added to the load indexes on tristate pins and adjusts the tristate load of reference library before comparing. Default: `-1`

Accepted values are:

`-1`	Follow setting of the adjust_tristate_load parameter.
`0`	Do not adjust the load indexes.
`1`	Add the `rise_capacitance` and `fall_capacitance` of the tristate pin to the associated load indexes.
`2`	The same as `1` except that the pin attribute `capacitance` will be added to the load indexes.
`21`	The same as `1` except that only timing arc loads will be adjusted (not power arc loads.)
`22`	The same as `2` except that only timing arc loads will be adjusted (not power arc loads.)

`-reltol <value | {list}>`

Specifies the relative tolerance limit for each comparison. Default: `0.01` (1%)

For more details, see the explanation of the `-abstol` option above.

`-report <filename>`

Specifies the filename with which the comparison report should be saved.
Default: `<comp_library_name>.cmp.txt`

An overall comparison summary is also written to the standard output.

`-report_pos_neg_max_diff`

Prints both negative and positive `max diff` and `diff%` in each table in the summary and verbose reports.

`-report_select_arc_nworst <pos_num>`

Generates a Tcl script with the select_arc command for the top `pos_num` worst absolute and relative outliers in each data table reported by `compare_library`.

Example:

```
compare_library -report_select_arc_nworst 1 $ref_lib
$comp_lib
```

Liberate generates file, `<comp_lib>_select_arc.tcl`, with `select_arc` command. If the `-report` option is also set with the `compare_library` command, the file path is printed in the file name as `<$report>_select_arc.tcl`.

This Tcl file contains the following content for arcs with outlier:

```
********************
Top 1 absolute positive outlier(s) in table 'trans'
select_arc -when "" -pin Z -pin_dir R -related_pin AN
-related_pin_dir F -type combinational CELLNAME


select_index -index_1 { 27 } -index_2 { 6 }


Line=696, diff=0.216569, diff%=541.42%, REF=0.04,
COMP=0.256569, nom_type=trans, nom_val=0.3667,
MC_sigma=N/A, sigma_type=N/A, MC_CI=N/A
********************
Top 1 absolute negative outlier(s) in table 'trans'
select_arc -when "" -pin Z -pin_dir F -related_pin IDDTN
-related_pin_dir R -type combinational CELLNAME


select_index -index_1 { 6 } -index_2 { 1 }


Line=1909, diff=-0.6219, diff%=-12.44%, REF=4.9998,
COMP=4.3779, nom_type=trans, nom_val=114.383,
MC_sigma=N/A, sigma_type=N/A, MC_CI=N/A


********************
```

```
Top 1 relative positive outlier(s) in table 'trans'

select_arc -when "" -pin Z -pin_dir R -related_pin AN
-related_pin_dir F -type combinational CELLNAME

select_index -index_1 { 18 } -index_2 { 5 }

Line=641, diff=0.181034333333, diff%=542.02%,
REF=0.0334, COMP=0.214434333333, nom_type=trans,
nom_val=0.3547, MC_sigma=N/A, sigma_type=N/A, MC_CI=N/A
********************
Top 1 relative negative outlier(s) in table 'trans'

select_arc -when "" -pin JTAGSAMP -pin_dir F -related_pin
PD -related_pin_dir F -type combinational CELLNAME

select_index -index_1 { 6 } -index_2 { 4 }

Line=4139, diff=-0.0505666666667, diff%=-67.06%,
REF=0.0754, COMP=0.0248333333333, nom_type=trans,
nom_val=101.665, MC_sigma=N/A, sigma_type=N/A, MC_CI=N/A
```

`-skip {`*`list`*`}` Specifies a list of data comparison types to skip. See the `-type` option for a list of supported types. Default: *none* (do not skip any comparison types)

`-type {`*`list`*`}` Specifies a list of data comparison types to include. Default: *all* However, the exception to the default behavior is that CCSP types must be specified explicitly; they are *not* included by default.

Valid comparison types are:

```
cap, ccs, ccs_cap, ccs_delay, ccs_trans, ccs_retain,
ccsn_dc, ccsn_prop, ccsn_vout, ccsp, ccsp_cap,
ccsp_dc, ccsp_lc, ccsp_res, constraint, delay, ecsm,
ecsm_cap, em, em_maxcap, hyper, leakage, max_cap,
max_trans, noise, power, retain, retain_trans,
si_prop_h, si_prop_w, siv, trans, setup, hold,
removal, recovery, min_period, mpw, nonseq_setup,
nonseq_hold, nochange, ocv_const, ocv_delay,
ocv_retain, ocv_retain_trans, ocv_trans,
ocv_const_mean_shift, ocv_const_stddev,
ocv_const_skewness, ocv_delay_mean_shift,
ocv_delay_stddev, ocv_delay_skewness,
ocv_trans_mean_shift, ocv_trans_stddev,
ocv_trans_skewness
```

In addition, a small collection of "macro-types" are available. These are convenient groupings of some of the basic types:

all = *(this is the default)*

capacitance = {cap ccs_cap ccs_retain ecsm_cap ecsm_cap_variation in_cap max_cap miller_cap}

constraint = {setup hold recovery removal mpw nonseq_setup nonseq_hold}

current = {ccs ccsn_dc ccsp siv}

timing = {delay delay_variation ecsm ecsm_variation max_trans time_const retain retain_slw trans trans_variation}

voltage = {hyper noise ccsn_vout}

The `compare_library` command separates dynamic power from hidden power. For example, specifying `-type {power}` compares the dynamic power, while specifying `-type {hidden_power}` compares the hidden power, and `-type {power hidden_power}` compares both.

`-unmatched` — Report unmatched data entries.

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

`-verbose` — Report all comparisons regardless of tolerances.

The generated report shows every comparison including those that did not exceed a tolerance.

<ref_lib> — Reference library.

<comp_lib> — Comparison library.

When comparing libraries, the data entries must have equivalent conditions. Two entries are deemed as equivalent if they have the same or overlapping logic conditions, related pins, and data type. In some instances not all the data in the reference library will have an equivalent in the comparison library. To report these entries, use the `-unmatched` option.

To compare entries only when there is an exact match in the *when* conditions, use the `-exact_match` option. If comparing libraries with different cell names, use the `define_map` command to map the names in the comparison library to the reference library.

Note: All the pin names must match.

Specify the `-multiple_matches` option to report comparison of all arcs that have functional overlap with a reference arc. The default is to report the table that gives the best match. Multiple arcs will be 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_match` option.

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

```
* : scaling due to slew thresholds or units
^ : input slews extrapolated
~ : output loads extrapolated
! : the indexes 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 will be compared. If the number of indexes (dimensions) differs between two data groups then 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 will be expanded to a 2-D slew/load table by using the first value of the load indexes from the 2-D table.

When the reference and comparison library values are 0 (including for cap max_tran, max_cap etc.) a report will be generated. If the reference value is zero and the comparison value is non-zero, then the percent difference is reported as a "/0". This point is not included in the overall average equation but it is counted as an outlier.

Example

```
# Set all relative tolerances to 2%, constraint tolerance
# to 3%, power tolerance to 5%.  Set absolute tolerance
# values for constraint, transition, leakage, and power
compare_library \
    -reltol { all 0.02  constraint 0.03 power 0.05 } \
    -abstol { constraint 5e-12
              trans 5.0e-12
              leakage 2.e-15
              power 3e-15 } \
    ref.lib \
    comp.lib
```

Sample Output Report

```
Legend : < outlier,  * scaled, ! indices switched, ^ slews extrapolated, ~ loads
extrapolated, + padding added, /0 divide by zero Legend : / slews interpolated, #
loads interpolated


*** 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 |       -2.1% |        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 |        -2.1% |        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 |
+----------+---------------+---------------+----------+


*** LIBRARY Comparison of comp.lib to ref.lib completed on Wed May 31 14:39:41 PDT
2006
```

Explanation of the Output Report Terminology

- Ref Library: The first library listed in the `compare_library` command
- Comp Library: The second library listed at the end of the `compare_library` command.
- Library and Cell Attribute Comparison groups:
  - Level: The level in the library where the attribute was found, either library level or particular to a cell.
  - Attribute: The name of the attribute.
  - Ref Value: The attribute value from the reference library Comp Value: The attribute value from the comparison library

- Data Comparison Table:
  - Row #: The row index number.
  - Pin Name: This field identifies the arc being compared. It can contain a pin, related_pin and their directions. If a single pin is listed then this is a hidden power or an MPW arc.
  - Ref Value: The data value as extracted from the ref (reference) library.
  - Comp Value: The data value as extracted from the comp (comparison) library
  - Diff: Ref_Value - Comp_Value
  - Diff %: Diff / Ref_Value
  - Type: The type of the data. Example: leakage, delay, rising, falling and power.
  - Index_1: The value of the index_1 in the Ref Library
  - Index_2: The value of the index_2 in the Ref Library

  Note: When the `index_1` and `index_2` values do not match, the right side of the row shows one of the following characters to indicate that extrapolation or interpolation occurred: ^, ~, /, or #. Refer to the legend provided at the top of the report for a detailed description.

- Data Comparison Summary:
  - Data Type: The type of the data.
  - Entries: The number of entries compared.
  - Avg Diff: The sum of the Diff values in the table divided by the number of entries in the table.
  - Avg Diff%: The sum of the Diff% values in the table divided by the number of entries in the table.
  - Sigma %: The square root calculated using the following formula:
    `Sqrt(Abs(((Sum of Diff%^2)/ #Entries) - (Avg Diff% ^2)))`
  - Outliers: This is the total number of outliers. An outlier is a data comparison where both diff > abstol and diff% > reltol for the data type. Each outlier in the report is indicated by a "<" character at the end of the row.

    If there are outliers, the report is enabled. Once enabled, the `Max Diff` and `Max Diff%` are extracted from all data comparisons of each type and not just from the outliers.

- `Max Diff` is the maximum of all the absolute difference value entries per data type.
- `Max Diff%` is the maximum of all the relative difference value entries per data type.

Note: If there are no outliers, the `Max Diff` and `Max Diff%` values are not reported.

Sample Summary Report

```
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 |
+----------+---------------+---------------+----------+
```

Explanation of the Summary Report Terminology:

- Entries: The total number of entries that were compared.
- Avg Diff%: See above
- Sigma %: See above
- Pass%: (#Entries - Outliers) / #Entries
- Outliers: See above

conv_ccs_to_ecsm

Converts the CCS current to ECSM voltage. In addition, writes the user-provided CCS data along with the relevant capacitance, ref_time, vdd, gnd, and pin_direction; and the converted ECSM data.

Syntax

```
conv_ccs_to_ecsm \
     -time <value> \
     -current <value> \
     -ecsm_thresh_list <list> \
     -cap <value> \
     -ref_time <value> \
     -vdd <value> \
     -gnd <value> \
     -rise <0 | 1> \
     -units <units>
```

Options

`-time <value>`	Specifies the CCS time values.
`-current <value>`	Specifies the CCS current values.
`-ecsm_thresh_list <list>`	Specifies a list of the ECSM waveform normalized thresholds. Default: `{0.02 0.05 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 0.95 0.98}`
`-cap <value>`	Specifies the capacitance value.
`-ref_time <value>`	Specifies the reference time value.
`-vdd <value>`	Specifies the power value.
`-gnd <value>`	Specifies the ground value.
`-rise <0 | 1>`	Specifies the pin direction.
`-units <units>`	Specifies the time, capacitance, and current units. Default: `"ns, pf, mA”`

Example

```
conv_ccs_to_ecsm \
    -time {3.82839004e-03, 4.02199989e-03, …, 2.03297995e-02} \
```

```
-current {6.80920035e-02, 7.60482997e-02, …, 8.31646976e-05} \
-cap 0.00063 \
-ref_time 0.00203718 \
-vdd 1.1 -gnd 0
```

conv_ecsm_to_ccs

Converts the ECSM voltage to CCS current. In addition, writes the user-provided ECSM time and threshold list along with the relevant capacitance, ref_time, vdd, gnd, and pin_direction; and the converted ECSM data.

Syntax

```
conv_ecsm_to_ccs \
     -time <value> \
     -ecsm_thresh_list <list> \
     -cap <value> \
     -ref_time <value> \
     -vdd <value> \
     -gnd <value> \
     -rise <0 | 1> \
     -units <unit>
```

Options

`-time <value>`	Specifies the ECSM time values.
`-ecsm_thresh_list <list>`	Specifies a list of the ECSM waveform normalized thresholds.
`-cap <value>`	Specifies the capacitance value.
`-ref_time <value>`	Specifies the reference time value.
`-vdd <value>`	Specifies the power value.
`-gnd <value>`	Specifies the ground value.
`-rise <0 | 1>`	Specifies the pin direction.
`-units <unit>`	Specifies the time, capacitance, and current units. Default: `"ns, pf, mA"`

Example

```
conv_ecsm_to_ccs \
    -time {1.98359019e-03, 2.23953067e-03, …, 1.14016142e-02} \
    -ecsm_thresh_list {0.02 0.05 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 0.95 0.98} \
    -cap 0.00063 \
    -ref_time 0.00203718 \
    -vdd 1.1 -gnd 0
```

copy_arc

Copies the data from one arc to another arc. This command is executed during model generation (see write_library) and affects the output library. Therefore, this command must be issued prior to model generation with `write_library`.

Options

`-from_cell <name>` Specifies the cell name to copy data from.

`-from_dir <name>` Specifies the data direction to copy from. For example, `"rise"` for `cell_rise` and `"fall"` for `cell_fall`. Default: `""` (copy both rise and fall data).

`-from_library <name>`

Specifies the library name to copy the arc from. Default: the first library read.

`-from_pin <name>` Specifies the pin name in the specified `-from_cell` to copy data from.

`-from_related <name>`

Specifies the related_pin timing to copy data from under the specified `-from_cell` and `-from_pin`. Default: `"*"` (copy arcs regardless of related pin)

`-from_related_pg <name>`

Specifies the related power/ground pin to copy data from. Default: `"*"` (copy arcs regardless of related power/ground pin)

`-from_timing_sense <value>`

Specifies the timing sense to copy data from. (Example: positive_unate)
Default: `""` (Copy regardless of timing sense)

`-from_timing_types {list}`

Specifies a list of timing types to copy data from. Default: `{}` (copy regardless of timing type)

`-from_subgroup_type <name>`

Specifies the name of a subgroup type to copy data from. For example: `"cell_rise"` or `"fall_transition"` for `-type delay`. Default: `""` (copy all sub-groups).

`-from_when <value>`	Specifies the `when` condition of the arc to copy data from. Default: `"*"` (copy all arcs regardless of the `when` condition)
`-margin {list}`	Specifies the absolute margin (in Seconds) to add to the copied data. You can use one value for all subgroups or a list of *<subgroup> <value>* pairs. Default: `0.0` (no margin) A single value will be applied to all tables in a timing group. If multiple values are specified there must be one value for each table in the timing group for the "from" arc. (Example: for a delay table, specify margin to add in this order: `rise_delay`, `fall_delay`, `rise_transition`, `fall_transition`.)
`-method < replace | append | augment >`	Specifies the copy method to use. The supported methods are: `"replace"`, `"append"`, and `"augment"`. Default: `"replace"` (replace the existing matching arc)
`-rel_margin {list}`	Specifies the relative margin to add to the copied data. You can use one value for all subgroups or a list of *<subgroup> <value>* pairs. For example, `{rise_constraint 0.1 fall_constraint 0.2}` ***or*** use `0.05` to represent a `5%` relative margin. Default: `0.0` (0%) With the `-rel_margin` option, you can specify a single value or a list of values (similar to margin.)
`-swap_direction`	Specifies to swap rise data with fall data and conversely after copying.
`-to_cell <name>`	Specifies the cell name to copy the data to. Default: see `-from_cell`

`-to_pin <name>` Specifies the pin name in the `-to_cell` to copy the data to. This option accepts a wildcard. Default: see `-from_pin`

For example, the following command enables a *one-to-many copy* operation by copying the relevant data selected using the `-pin`, `-from_related`, and so on options to all pins that match the wildcard name:

```
copy_arc \
    -type {delay} \
    -method replace \
    -from_cell $cell \
    -from_pin SLEEPOUTN \
    -from_related SLEEPN \
    -from_timing_sense positive_unate \
    -to_cell $cell \
    -to_pin *_altos* \
    -to_related SLEEPN \
    -to_timing_sense negative_unate
```

`-to_related <name>` Specifies the related_pin to copy the data to. Default: see `-from_related`

`-to_related_pg <name>`

Specifies the related power/ground pin to copy the data to. Default: see `-from_related_pg`

`-to_subgroup_type <name>`

Specifies the name of subgroup type to copy the data to. For example: `"cell_rise"` or `"fall_transition"` for `-type delay`. Default: `""` (see `-from_subgroup_type`)

`-to_sdf_cond <string>`

Specifies the `sdf_cond` string to use after copying the arc. Default: Create the `sdf_cond` from the `-to_when` condition.

`-to_timing_sense <value>`

Specifies the timing sense to copy the data to. Default: `""` (see `-from_timing_sense`)

`-to_timing_types {list}`

Specifies the timing type(s) to copy the data to. Default: `{}` (see `-from_timing_types`)

`-to_when <value>`	Specifies the `when` condition to use after copying the arc. Default: see `-from_when`

`-type {list}`	Specifies the type of data to be copied. Default: copy all data types.
	
	The supported types are: `cap`, `ccs`, `ccsn`, `ccs_retain`, `constraint`, `delay`, `em`, `em_maxcap`, `hold`, `leakage`, `mpw`, `power`, `recovery`, `removal`, `retain`, `setup`, and `timing`.
	
	The `cap` value allows copying all the capacitance related data from one pin to another. This includes capacitance attributes such as `capacitance`, `rise_capacitance`, `fall_capacitance` as well as any CCS or ECSM receiver capacitance. For example:
	
	`copy_arc \`
	`-type cap \`
	`-from_pin "my_D" -to_pin "your_D\`

Note: The `copy_arc` command does not currently support `write_socv`.

create_library_index

Creates a Tcl index file in the specified index directory.

To ensure that an index file exists before the library is read, the `create_library_index` command must be run before using the read_library command with the `-cell` option. This allows Liberate to read libraries much quicker because the cells are extracted faster without any impact on the disk space.

Use the `create_library_index` command with the `read_library -cell $cell` command before commands such as interpolate_library that run faster when cell-level libraries are involved.

Syntax

```
create_library_index \
     -directory <index_directory> \
     <list_of_input_libs>
```

Options

`-directory <index_directory>`

Specifies the index directory where the Tcl index file should be saved.
Default: `"./index"`

`<list_of_input_libs>`

Specifies a list of input libraries.

create_task

Creates a task in the task list that is to be later run in parallel on the clients. Each task runs the specified script with the given argument list. A log file for each task is created under the specified log directory. This command should be used together with the parallelize_tasks command in the packet arc flow with Bolt job distribution system.

Note: To parallelize different tasks that are dependent on each other, you can use the `create_task` and `parallelize_tasks` command sets multiple times in the same session of Liberate. This helps to leverage the Bolt job distribution system and reduce the TAT by parallelizing the task at per PVT or per cell level and utilizing multiple CPUs instead of just one.

Options

`-script <script_name>`

(Mandatory) Specifies the name if the script that runs the task.

`-args <arguments_list>`

(Optional) Lists the arguments to be passed to the script.

`-logdir <log_directory_name>`

(Optional) Specifies the directory where the log file for this task should be saved. When specified, a log file directory hierarchy is created; otherwise, the log files are saved under the current run directory.

Example

In `run.tcl` that creates tasks to run interpolate_library on cell-level libraries:

```
# setup bolt
set_var packet_arc_job_manager bolt
set_var packet_clients 5
set_var rsh_cmd "bsub -q liberate -R \"(OSREL==EE50||OSREL==EE60) rusage\[mem=1\] span\[hosts=1\]\" -o %B/%L -e %B/%L"
set_var packet_client_health_checks 1

# create tasks to run in parallel
set voltages {0.95 1.0 1.05}
foreach voltage $voltages {
    foreach cell $cells {
```

```
        # for each voltage/cell combination, create a task running
        interpolate_library.tcl with the given argument list
        create_task -script $rootdir/interpolate_library.tcl -args [list $rootdir
        $voltage $cell] -logdir $voltage/$cell
    }
}
```

define_arc

Specifies a user-defined arc to override automatic arc determination by Liberate. 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.

When this command is used without the `-ignore` option, the pin directions must be specified using a vector or the appropriate `define_arc` options such as `-pin_dir` and `-related_pin_dir`.

Options

`-attribute {`*`list`*`}` List of user-defined attribute/value pairs. These attribute and values pairs are added to the .lib for the specific arc.

For example:

```
define_arc \
-attribute {attribute_name1  <value>  \
            attribute_name1  <value>  \
            ....
            ...
          attribute_nameN  <value> } \
<cell_name>
```

`-ccsn_stage <arc | pin>`

Specifies the CCSN data type. Default: `pin`

`-ccsn_probe <`*`node`*`>` Specifies the CCSN internal probe node name used as the ccsn stage input/output.

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

Defines a list of four delay percentage measurement points (a ratio of VDD normalized to between 0 and 1) for the arc. For the `-delay_threshold` option, these values represent the following measurement thresholds in the exact order as given here: `input_rise_delay`, `input_fall_delay`, `output_rise_delay`, `output_fall_delay`.

If not specified, all delays are measured at the values defined by the `delay_*_*` parameters.

`-dependent_load {<`*`pin load_value`*`>...}`

List of pin-load pairs to add to dependent side pins. (See set_dependent_load)

This option provides user control over the load applied to side outputs that impact the arc. These are specified as a list of pin-load pairs, that is, `{pin1 load1 pin2 load2 ...}`. Dependent loads in the `define_arc` command will supersede 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.

Note: 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.

This option can have one of the following values:

`U` Sets a direction of up.

`D` Sets a direction of down.

`B` Sets a direction of both.

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

`-dual_related <`*`name`*`>`

Specifies the name of the other pin in a pair of differential input pins.

When differential pairs are specified for inputs using `-dual_related` or for outputs using `-dual_pin`, the delay measurements can be made using the voltage crossover between the differential signals. To request that the delay measurements use the crossover point, the `-delay_threshold` option must be specified with a value of `cross` instead of a ratio. For example:

```
-delay_threshold { 0.5 0.5 cross cross }
```

`-extsim_deck_header`

Allows to provide external simulator commands directly to the external simulator on an individual arc basis without using the Liberate process or reviewing them. This option is intended to be used when an external simulator is used (refer to the `-extsim` option of the `char_library` command). It is a local arc specific version of the extsim_deck_header parameter. 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 at the top of simulation deck. For example:

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

`-ic <"ic list">` Defines the initial conditions to be applied during simulation. There should be one voltage value for each pin in the pin list.

`-ignore` Prevents characterization 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 not required and will be ignored. This option can be used to disable the internal view in Liberate from analyzing the specified arc.

`-load_dir <U | D | B>`

Specifies whether the pull-up resistance, the pull-down resistance, or both resistances should be applied to this arc.

This option can have one of the following values:

- `U` Applies the pull-up resistance.
- `D` Applies the pull-down resistance.
- `B` Applies both the pull-up and pull-down resistances.

`-logic_condition <value>`

Provides the logic states for static side pins. This option uses the same syntax and affects vector selection in the same manner as the `-when` option. However, unlike the `-when` option, it does not appear in the output library. This option can be used with or without the `-vector` option.

Note: The `-when`, `-logic_condition`, and `-vector` options must be consistent; otherwise, the `define_arc` command will be ignored. This mean that a given pin must have either 0 / X or 1 / X in all three options.

`-margin`

Allows a user-specified backoff margin to be added on a per-arc basis. This only applies to timing constraint arcs characterized using the path-delay method, such as `define_arc` commands that use the `-pin_probe` and `-related_probe` options and `set_constraint` commands with the `-pin_probe_factor`, `-related_probe_factor`, `-min_margin`, and `-max_margin` options.

`-metric { list }`

Specifies a list of one or more metric types to use for measuring timing constraints. For related details, see Using the -metric option.

Note: When a list of metrics is used in conjunction with a list of `metric_thresh` values, the order and number of the parameters and values must match between these lists.

For example:

```
define_arc  \
    -metric {
        delay
        constaint_delay_degrade_abstol
        constraint_delay_degrade_abstol_max } \
    -metric_thresh {
        0.1
        1e-12
        1e-12  }  \
    -pin D \
    -related_pin clock \
    -vector { RR ... } \
    myCell
```

`-metric_thresh {`*`list`*`}`

Accepts a list of threshold values. When used in combination with the `-metric` option, overwrites the values specified by the set_constraint_criteria command, or by the corresponding global parameter, constraint_slew_degrade, for this arc. This list must have one value corresponding to each criteria in the metric list.

Note: When a list of metrics is used in conjunction with a list of `-metric_thresh` values, the order and number of the parameters and values must match between these lists.

`-no_harness` Specifies that the harness must not be included in the simulation deck.

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

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

Specifies a list of pins corresponding to vector.

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

This option can have one of the following values:

`R` Specifies a rising transition.

`F` Specifies a falling transition.

`-pin_gnd {`*`pin voltage ...`*`}`

Specifies a list of arc-specific input gnd pin and voltage pairs.

Note: This option can be used without the `-vector` option.

`-pin_load <`*`template`*`>`

Specifies additional circuitry to be applied to all the destination `pins` of the `define_arc` command. This 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 pull-up and pull-down resistances and series resistance.

`-pin_load_dir {pin_name load_template direction}`

Applies specific pin loads to specific output pins. This provides more granularity of control than the `-pin_load` option. This option takes a triplet of arguments: `pin_name`, `load_template`, and `direction`. It can also be specified as a list of triplets. Here, the `load_template` argument is the name of the loading configuration defined with the `define_pin_load` command, and direction is specified with `U` (pull-up resistance), `D` (pull-down resistance), or `B` (both pull-up and pull-down resistances).

Note: `-pin_load_dir` overrides any `-pin_load` that might be present. For example:

```
-pin_load_dir {pin1 load_template1 U pin2 load_template2 D}
```

`-pin_probe {pins}` Specifies a list of pin monitor node names.

`-pin_probe_dir {pins}`

Specifies a list of pin monitor node directions `[ R | F ]`.

`-pin_probe_threshold {pins}`

Specifies a list of pin monitor node thresholds.

`-pin_vdd {pin voltage ...}`

Specifies a list of arc-specific input vdd pin and voltage pairs.

Note: This option can be used without the `-vector` option.

`-pg_pin <value>` Assign a value to this power/ground pin only

`-prevector_pinlist {list}`

User-specified pin list for pre-vectors.

`-prevector_slew`

Specifies the slew rate to apply to the input and bidi pins in the `-prevector_pinlist`. Supported values are: `<value>`, `min`, `mid`, `max`, `index_n`. For more information, see prevector_slew.

`-prevector "<vector ... >"`

User-specified initialization vectors. Default: no prevectors

For related information, see Using the -prevector option.

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

Defines the node(s) to monitor when determining the timing constraint. It can be an external pin, such as, the `Q` pin in a flip-flop or an internal node name.

Use the `-probe altos_internal` option when a constraint can be measured at both an internal node and an output pin. This instructs Liberate to use the internal probe node. If the `probe` argument is not specified, the pin defined by the constraint_output_pin parameter is probed.

This option can have one of the following values:

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

`altos_internal`

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

`-probe_dir <R | F>` Specifies the node direction that the probe nodes will be transitioning. The acceptable values are: `R` and `F`.

One direction can be provided for each probe node.

Note: The `-probe_dir` option is used with the `-probe` option.

`-probe_trigger <`*`pin`*`>`

Specifies the input pin to use as trigger for the trigger-to-probe delay measurement for cases where the *Inside View* algorithm cannot determine the correct pin to use.

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

Specifies a 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 pins.

This option can have one of the following values:

`R` Specifies a rising transition.

`F` Specifies a falling transition.

`-related_probe {`*`pins`*`}`

Specifies a list of related monitor node names.

Note: The `-pin_probe` and `-related_probe` options are used to specify the measurement target nodes.

`-related_probe_dir {R | F}`

Specifies a list of related_probe monitor node directions `[ R | F ]`.

`-related_probe_threshold {`*`pins`*`}`

Specifies a list of related monitor node thresholds.

`-sdf_cond <"`*`function`*`">`

Defines logic conditions of the other pins of the cell to enable the arc using SDF compatible syntax.

Note: If `sdf_cond` is specified by the user using the `define_arc -sdf_cond` command, Liberate will leave it intact and perform no replacement. A warning message will be displayed indicating that the SDF conditions have been overwritten by user when the `write_library` command is run. Any special characters (that is, " +|&*") will be kept.

Sample message:

```
*Warning* (write_library): The sdf_cond has been
overridden by user supplied values (see define_arc -
sdf_cond) in one or more timing groups. These user
supplied values are not checked for consistency by
Liberate.
```

If `msg_level >0`, Liberate will dump the details for ***all*** cells that contain user-defined arcs with `sdf_cond`.

`-slew_threshold { lower_rise upper_rise lower_fall upper_fall }`

Defines a list of four slew percentage measurement points (a ratio of VDD normalized to between 0 and 1) for the arc. For the `-slew_threshold` option, the values in the list represent the following measurement thresholds in the exact order as given here: `lower_rise`, `upper_rise`, `lower_fall`, `upper_fall`.

If not specified, 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` parameters.

`-type < async | ccsn_first | ccsn_last | combinational | cin | dc_current | disable | edge | enable | hidden | hold | max_clock_tree_path | min_clock_tree_path | min_period | minperiod | mpw | nochange_high_high| nochange_high_low | nochange_low_high | nochange_low_low | non_seq_hold | non_seq_setup | power | recovery | removal | setup >`

Specifies the type of arc. Default: `combinational`

For related details, see Using the -type option.

`-value <value | {list}>`

Overrides the characterized values. Default: use the characterized values.

The values override the characterized results and force a value (or list of values) for all entries into the data table for the specified arc.

This option accepts a single value or a list of values. If a single value is provided, then the table gets a scalar type of data. However, if a list is specified, there must be one entry for each point in the data table. For example, a 7x7 table would require 49 values. For any time-based arc, the value is in seconds (that is, 5e-9 = 5ns).

`-value_trans <value | {list}>`

Overrides the values in the transition table. Default: use the characterized values.

The `-value_trans` option works similarly to the `-value` option except that it applies to transition table. See the `-value` option for more information.

`-vector <"stimulus">`

Specifies the vector stimulus used to simulate the specified arc, each bit can be one of: `R`, `F`, `X`, `1`, or `0`. For related details, see Using the -vector option.

`-when <"function">` Defines 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.

The `-when` option supports bus expansion only when there are the following types of operators present:

- `!` and `&` (exclamation mark and ampersand)
- `*` and `|` (asterisk and pipe symbol)
- `+` (plus)

Limited support for a ' (single quote) operator is also provided.

See Using the -when Option for an example.

`{cell_names}` (Required positional option) Specifies a 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.

Liberate can measure the path from the `pin` to a user-defined `pin_probe` and from the `related_pin` to a user-defined `related_probe`. The `pin_probe` and `related_probe` options can be used to specify the measurement target nodes. The `-pin_probe_dir`, and `-related_probe_dir` options can be used to specify the transition direction of the `pin_probe` and `related_probe` nodes. The `-pin_probe_threshold` and `-related_probe_threshold` options specify a ratio of supply and can be used to specify the measurement thresholds for the `pin_probe` and `related_probe` nodes. See Constraint-related Delay and Clock Path Measurement for more information and an example.

The define_arc command can be used to guide the CCSN characterization. Liberate attempts to automatically identify internal probe nodes. Use the ccsn_probe option to identify a particular node to use as the probe. When arc based ccsn is specified, the same node must be used as the probe node so a ccsn_first_stage and ccsn_last_stage are consistent and represent an arc from the input to the output pin. For example:

```
define_arc \
        -related_pin A \
        -type ccsn_first \
        -ccsn_stage pin \
        myCell
```

This command must be used before the char_library command is run.

Using the -metric option

The -metric option is used for setting constraint criteria for timing constraint arcs. It can contain a list that includes the following keywords corresponding to Liberate parameters:

```
delay | delay_both_edges | constraint_delay_degrade
glitch | constraint_glitch_peak
slew | constraint_slew_degrade
width | constraint_width_degrade
constraint_delay_degrade_abstol
constraint_delay_degrade_abstol_max
constraint_delay_degrade_minimize_dtoq_tol
constraint_delay_degrade_minimize_dtoq
path_delta
removal_glitch_peak
```

Note: path_delta cannot be used together with other metrics in the same define_arc command.

- When set to delay, the arc will be in violation when a delay change at the probed pin exceeds the metric threshold corresponding to the constraint_delay_degrade parameter.
- When set to slew, the arc will be in violation when the slew change at the probed pin exceeds either: the constraint_slew_degrade parameter; the threshold set with the -metric_thresh option; or the threshold set with the set_constraint_criteria command.
- When set to delay_both_edges, the metric applies to constraints where the probe node transitions in response to both the related and the constrained pins. The constraint is tested with delays from both the constrained and related pins to the

associated transition on the probed node. Degradation failure on either edge indicates a failure of the constraint.

- When set to `glitch` the arc will be in violation when the glitch-peak at the probed pin exceeds constraint_glitch_peak parameter.
- When set to `width`, the arc will be in violation when the output pulse width degrades beyond the percentage set in either: the constraint_width_degrade parameter; the threshold set with the `-metric_thresh` option; or the threshold set with the set_constraint_criteria command.
- If both delay and glitch criteria is set for hold time, the glitch crteria is used.
- The `-metric path_delta` option causes Liberate to measure the data path, (path from the `pin` to the `pin_probe`) and the clock path (the path from the `related_pin` to the `related_probe`) and report the difference between these two paths as the constraint value. For more information, see Appendix B, "Constraint-related Delay and Clock Path Measurement".

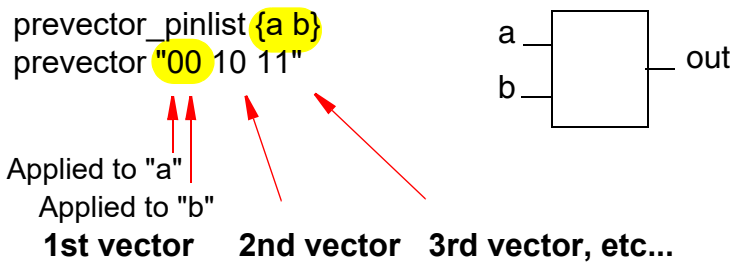
  Example:

```
define_arc \
   -type setup \
   -metric path_delta \
   -pin D -pin_dir R \
   -pin_probe n1 -pin_probe_dir R \
   -related_pin clk -relate_pin_dir R \
   -relate_probe n2 -related_probe_dir F \
   { myFF }
```

Using the -prevector option

The `-prevector` option specifies an arbitrary simulation stimulus to be applied before the vector where the characterization will be measured. Pre-vectors are used to put a device in a user-defined state before proceeding with characterization. Prevector values must be `0` or `1`. See related parameters: prevector_period and prevector_slew.

This option must be used in conjunction with `-prevector_pinlist`. The following illustrates how prevectors are specified with the `-prevector_pinlist` option:



Important

1. The last state of the prevector must match the beginning state of the vector for a given pin, otherwise an unintended transition will occur.

2. When `-prevector` is specified, Liberate disables the *Inside View* algorithm for this arc and operates as if it is in `io` mode for this arc. In `io` mode, `probe` is required for all constraint arcs, including `mpw`. (See char_library)

3. It is recommended that only one pin in the `-prevector_pinlist` transition should be switching at a time in the prevector. This helps to avoid race conditions that can cause the prevector to produce incorrect or undesired initial values.

Example 1:

```
# Define the IOCELL
define_cell \
  -input { D } \
  -output { Q } \
  -clock { CK } \
  -pinlist { D CK Q } \
  -delay delay_template_3x3 \
  -power power_template_3x3 \
  MYCELL

define_arc \
  -prevector_pinlist {D CK} \
  -prevector "00 01 00 10" \
  -vector    "1RR" \
  -related_pin CK \
  -pin Q \
  MYCELL
```

Example 2:

```
# constraint arcs from CK => CK using prevector min_pulse_width
define_arc \
       -type min_pulse_width \
       -prevector_pinlist {D CK Q} \
       -prevector {100 111 001} \
       -vector {0RF} \
       -related_pin CK \
       -pin CK \
```

```
        DFF
# constraint arcs from CK => CK using prevector min_pulse_width
define_arc \
        -type min_pulse_width \
        -prevector_pinlist {D CK Q} \
        -prevector {100 111 011} \
        -vector {0FF} \
        -related_pin CK \
        -pin CK \
        DFF
```

Using the -type option

The `-type` option defines the type of arc. The arc can be a combinational path from input pins (`related_pin`) to output pins (`pin`) for combinational cells. It can also be a timing constraint of type *setup*, *hold*, *recovery*, *removal and mpw* between data (`pin`) and a clock (`related_pin`) for sequential cells. Other valid types include `async`, `cin`, `combinational`, `disable`, `edge`, `enable`, `hidden`, `non_seq_hold`, `non_seq_setup`, and `power`**.** An `async` arc corresponds to a preset or clear transition. An `edge` arc between an input and an output pin will be an edge-triggered transition. The `enable` and `disable` types are used for specifying arcs that enable and disable tristate gates. A `hidden` arc specifies an `internal_power` group found under the `input` or `bidi` pin that does not cause any output to transition and is used to characterize the hidden power for that pin. The `non_seq_hold` and `non_seq_setup` types are used for specifying setup and hold arcs between a pin and a non-clock related pin. The `power` arc specifies an `internal_power` group found under the `output` or `bidi` pin where one or more outputs transition. The `dc_current` type is used primarily for power switch cells.

For `define_arc -type power`, Liberate does not forget the input pin direction at the time of characterization. If both directions are allowed and simultaneous switching is required, the switching of direction of all pins is consistent with the requirements of `define_cell -when`.

Use the `cin` arc type to specify the conditions to be used for characterizing the input capacitance for a pin. This should only be needed when there are no arcs characterized originating from the specified pin. If the cell has a harness specified (see `define_cell -harness`), the harness will automatically be disconnected for the `-type cin` arcs.

A two-dimensional Minimum Pulse Width (MPW) template (see `define_template -type mpw`) requires an output pin as the probe when using the `char_library -auto_index` option. This is because the automatic scaling of the load capacitance for the MPW arc depends on a delay arc corresponding to the mpw arc. When `define_arc -type mpw` is set, the output load of the corresponding delay arc is scaled automatically and then applied to the load index of the MPW arc. The pin of the delay arc must match the probe of the mpw

arc, and the related_pin of the delay arc must match the pin of the MPW arc. If the probe of the MPW arc is empty or is an internal node, the program cannot find a corresponding delay arc and cannot apply auto_index scaling to the MPW load capacitance.

Specify the define_arc commands for nochange arcs whenever they are required to be modeled in the output library. These arcs will either have a user-defined value or be characterized by Liberate. See the parameters nochange_mode and nochange_value for more details. If nochange_*_* arcs are specified without -pin_dir or -vector and nochange_mode is set to 1, both rise_constraint and fall_constraint are characterized. If -pin_dir or -vector are specified only the specified rise_constraint and fall_constraint are generated. The nochange pull-in constraints use the same metric settings as delay: constraint_delay_degrade, constraint_delay_degrade_abstol, constraint_delay_degrade_abstol_max. Using abstol makes the most sense for nochange arcs because the object is to preserve the clock shape within some tolerance and not to locate the metastable region. All values should be positive.

When using a glitch metric for nochange arcs, the standard glitch metric settings apply.

If there is no define_arc -type nochange_*, no such arc is written to the output library. Nochange arcs are only generated if a define_arc exists.

Important

The define_arc command must be specified if Liberate is to characterize and model nochange arcs.

Example 1:

```
# Generate all 4 nochange arc constraints between EN and CKI with the correct
# metric (delay, pull-in, or glitch) for the specified cell (ANDX1 in this example).
set_var nochange_mode 1
define_arc -type nochange_low_high -pin EN -related_pin CKI ANDX1
define_arc -type nochange_high_high -pin EN -related_pin CKI ANDX1
```

Example 2:

```
define_arc \
    -type dc_current \
    -pin en \
    POWER_SWITCH
...
char_library -cells $cells -user_arcs_only -extsim spectre
```

Using the -vector option

The `-vector` option defines 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 pin list 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 buses 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.

All stimulus vectors that satisfy the `when` condition will be enumerated and simulated.


Caution

Do not define a vector that drives a node such that it cannot be resolved (that is, a given node is driven both high and low.) This is called a "collision", resulting in the node voltage becoming unknown and causing high leakage. If this occurs, Liberate ignores the define_arc command that caused this condition and outputs a warning message.

Examples:

```
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
```

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_arc \
    -type hidden \
    -when "OEN * !PAD" \
    -vector {R10X} \
    -pin IN \
    IOCELL
define_arc \
    -type hidden \
    -when "OEN * !PAD" \
    -vector {F10X} \
    -pin IN \
    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

# For differential output signals into a cell
# I -> PAD
# Pinlist : I PAD PADN
define_arc \
```

```
-vector "FFR" \
-delay_threshold {0.5 0.5 CROSS CROSS} \
-related_pin I \
-dual_pin PADN \
-pin PAD \
OCELL_diff
```

Using the -when Option

Examples

`define_arc -when {A[1:0] & B[1:0]} will be expanded to define_arc -when {(A[1] & A[0]) & (B[1] & B[0])}`

`define_arc -when {!A[1:0] & !B[1:0]} will be expanded to define_arc -when {(!A[1] & !A[0]) & (!B[1] & !B[0])}`

`define_arc -when {!A[1:0] & B[1:0] | !C[1:0]} will be expanded to define_arc -when {(!A[1] & !A[0]) & (B[1] & B[0]) | (!C[1] & !C[0])}`

Note: The support for operator ' is limited. For example, `define_arc -when {!A[1:0] & B[1:0]'}` will expand to `define_arc -when {(!A[1] & !A[0]) & (B[1]' & B[0]')}`. However, if there is a space between `]` and `'` (for example, `B[1:0] '`), the bus is wrongly expanded to `(B[1] & B[0]) '`. For example, `define_arc -when {!A[1:0] & B[1:0] '}` will wrongly expand to `define_arc -when {(!A[1] & !A[0]) & (B[1] & B[0]) '}`.

define_bundle_pins

Treats the pins as if they were from equivalent latches or flip-flops in a bank instead of including each of the input and/or output pins. For example, if pins `SE1` and `SE2` are from separate flip-flops that are identical, the bundle syntax merges them together in a single bundle with shared timing and power models. The resulting model is more compact than one in which all pins are uniquely specified. By using this method, all cells share a single model. It should only be used if each cell in the bank is identical.

Options

`-criteria {<delay | power> <min | avg | max>}`

Specifies how the characterized data is to be modeled when the `-use_pin` option is selected. Using the `-patterns` option restricts the bundle patterns considered for the selection. Without `-use_pin` all the bundle pins are measured separately. The `-criteria` option then controls how the individual pin models are combined into the bundle model. It does not affect the characterized vectors.

`-no_model`

Disables modeling of bundles in the output library. This option allows a bundle to be defined which is not modeled in the Library. This allows `-patterns` to be associated with the bundle pins to restrict characterization vectors for multi-bit cells without affecting the structure of the resulting library.

`-patterns { list }`

List of string patterns. This option specifies a list of patterns consisting of 0 and 1 to be assigned side pins that are members of the bundle or the string `all`. Side pins are the pins that do not logically impact the cone of logic identified by mega-mode. These patterns reduce the number of vectors to be simulated by restricting the settings of nodes which are not critical to a particular measurement. The default patterns are {0 1} which will create 2 vectors with all 0's and all 1's assigned to side pins in the bundle. Setting the patterns to `all` does not restrict the bundle pins in any way. Setting the patterns to {01 10} would use two patterns with alternating 0's and 1's. The pattern is repeated as many times as necessary to cover the number of bits in the bundle so it is not necessary to devise a different pattern set for each bundle. As many patterns can be supplied as desired but increases the number of vectors. When `-patterns` is not set to `all` , the when conditions generated for the library does not include bundle pins. See (mega_mode_constraint, mega_mode_delay, mega_mode_hidden, and mega_enable).

`-use_pin <pin_name>`

Specifies the name of the pin used to acquire all timing and power models related to a bundle. This saves considerable cell analysis and simulation time, because arcs involving all other pins in the bundle will be ignored.

Note: If this option is used with `define_bundle_pins`, the command must be used ***before*** `char_library`.
If this option is not used, `define_bundle_pins` can be used after `char_library`, but before `write_library`.

`{cellNames}`	List of cell name (Required). Default: `{}`
`<bundle_name>`	The name of the bundle (Required). Default: `""`
`{pins}`	List of pins in the bundle (Required). Default: `{}`

Examples

```
define_bundle_pins -use_pin D1 $cellname D {D4 D3 D2 D1}
define_bundle_pins -use_pin Q1 $cellname Q {Q4 Q3 Q2 Q1}
```

define_bus

Groups the bus bits into a bus in the output library. It is not required if buses are explicitly defined in **define_cell**.

Options

`-by <integer>`	Specifies the bus bit increment. Default: `1`
`-dont_overwrite`	Specifies not to overwrite the previous bus definition. Default: overwrite (multiple `define_bus` commands for the same bus will be merged)
`-from <integer>`	Specifies where the bus starts. Default: `0`
	Note: The value specified with the `-from` option can be higher or lower than the `-to` option, but not equal. These option define the bus start and ending range.
`-patterns { list }`	List of string patterns. This option specifies a list of patterns consisting of 0 and 1 to be assigned to the side pins that are members of the bundle or to the string `all`. Side pins are the pins that do not logically impact the cone of logic identified by the mega mode. These patterns reduce the number of vectors to be simulated by restricting the settings of nodes that are not critical to a particular measurement. The default patterns are {0 1} that creates 2 vectors with all 0s and all 1s assigned to side pins in the bundle. Setting the patterns to `all` does not restrict the bundle pins in any way. Setting the patterns to {01 10} would use two patterns with alternating 0s and 1s. The pattern is repeated as many times as necessary to cover the number of bits in the bundle, so it is not necessary to devise a different pattern set for each bundle. As many patterns can be supplied as desired, but it increases the number of vectors. When `-patterns` is not set to `all`, the `when` conditions generated for the library do not include bundle pins. See mega_mode_constraint, mega_mode_delay, mega_mode_hidden, and mega_enable.
`-to <integer>`	Specifies where the bus ends. Default: `0`
`-use_bit`	Specifies the bus bit whose characterized data will be applied to the entire bus. Default: the value of the `-from` option
<cellName>	List of cell names (Required). Default: `{}`
<bus_name>	Specifies the bus name (Required). Default: ""
	Note: The `bus_name` option should be the name of the pins minus the bus index.

This command must be used before `write_library` and can also be used after

`char_library` or `read_ldb`.

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 amongst a group of cells.

Note: For variation characterization in Liberate Variety or in unified characterization flow of Liberate Trio, the minimum dimension of a template is 2x2. Templates smaller than 2x2 can be defined and stored in a library, but cannot be used for variation characterization.

Options

`-async {pin_names}`	List of asynchronous pin names.
`-bidi {pin_names}`	List of bi-directional pin names.
`-clock {pin_names}`	List of clock pin names
`-constraint <name>`	Name of template for constraint tables. The specified template is used 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, the construct is not characterized. The `-constraint` option enables characterization of timing constraints (setup, hold, recovery, removal). The range of input slews to use for the data and clock signals is defined by the given template name where the template is predefined using the `define_template` command.
`-delay <name>`	Name of template for delay tables. The specified template is used 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, the construct is not characterized. The `-delay` option enables characterization of cell delay and output slew for the non-linear delay model (NLDM). The range of input slews and output loads to use for this construct is defined by the given template name where the template is pre-defined using the `define_template` command.

`-harness <load_subckt_name>`

Name of subcircuit containing custom load.

The `-harness` option allows you to specify an arbitrary SPICE format circuit to *wrap* around the cell that is being characterized. You need to provide a subcircuit containing an arbitrary input circuit and/or an arbitrary output load. The harness subcircuit must have been loaded with `read_spice`. The harness subcircuit must be defined with pins that have the identical names as the port names of the cell being characterized since the harness pin names are name mapped to the subcircuit being characterized. The following table indicates a naming convention used to refer to connections from the original characterization setup that should be shifted to new connections in the harness (see example):

Harness Parameter Description	**Harness Name**
Input stimulus	`<`*`subckt_pin`*`>_altos_stim`
Cell input port	`<`*`subckt_pin`*`>_altos_in`
Cell output port	`<`*`subckt_pin`*`>_altos_out`
Output load	`<`*`subckt_pin`*`>_cap`
Supply voltage	`<`*`subckt_pin`*`>_voltage`

`-ignore_input_for_auto_cap {`*`pin_names`*`}`

List of input/bidi pins. Any timing arcs that originate at the pins specified in this list will not be considered for `auto_index` and `auto_max_capacitance` calculations. Instead, the values specified in the templates (see define_template) for this cell will be used. This option should not be used with `scale_load_by_template`, `scale_tran_by_template`, and `-auto_index` option of the write_template command because the template values will not represent correct values.

`-ignore_output_for_auto_cap {`*`pin_names`*`}`

List of bidi/output pins or an asterisk (`*`) to match all output pins.

Any timing arcs that terminate at the pins specified in this list will not be considered for `auto_index` and `auto_max_capacitance` calculations. Instead, the values specified in the templates (see `define_template`) for this cell will be used. This option should not be used with `scale_load_by_template`, `scale_tran_by_template`, and `-auto_index` option of the `write_template` command because the template values will not represent the correct values.

`-ignore_pin_for_ccsn {`*`pin_names`*`}`

List of input/bidi pins. No CCSN data is characterized for the specified pins.

`-input {`*`pin_names`*`}` List of input pin names.

`-internal {`*`pin_names`*`}`

Defines a list of pin that are internal to the cell, that is, they are not part of the port list of the top-level subcircuit definition for the cell. Such pins must be defined when they are used as `-pin` or `-rel pins` option of the `define_arc` command.

Note: This option is available only in Liberate and Liberate AMS.

Arcs are not generated automatically to and from the internal pins. However, when you specify the internal pins with the `define_arc` command, the internal pins and the corresponding generated arcs are reflected in the `.lib`. During modeling, all arcs generated for the internal pins are forced to 1D format.

`-internal_supply {`*`supply_names`*`}`

List of switched supply pin names.

The `-internal_supply` option is used for cells such as power switch cells to identify output pins that are to be treated as switched power nets. The internal_supply net must be a port in the subckt definition of the cell. When this option is used, Liberate will characterize additional dc_current data for CCCs connected to the specific internal supply pin(s), post process the dc_current groups into the correct format recognized by LC, and output the power-down function attribute. All internal_supply pins must also be identified as a supply using the set_gnd commands.

`-mpw <`*`name`*`>` Name of template for MPW tables.

The `-mpw` option enables characterization of a two dimensional MPW table. The range of input slews to use for the data and clock signals is defined by the given template name where the template is pre-defined using the `define_template` command.

`-output {`*`pin_names`*`}`

List of output pin names.

`-pad {`*`pin_names`*`}`

List of pins that are pads. These pad pins must also be included in the associated type of pin definition specified using the `define_cell` command options, such as `-input`, `-output`, `-bidi`, and so on.

`-pinlist {`*`pin_names`*`}`

Defines the pin order and 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.

`-power <name>`

Name of template for power tables. The specified template is used 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, the construct is not characterized.

The `-power` option enables characterization of switching power and hidden power (power dissipated when the output doesn't switch). The range of input slews and output loads to use for this construct is defined by the given template name where the template is pre-defined using the `define template` command.

`-scan {pin_names}`

List of scan related pin names

`-scan_cell_postfix "string"`

String added to end of dummy scan cell and footprint names

`-scan_cell_prefix "string"`

String added to beginning of dummy scan cell and footprint names

`-scan_disable {pin value, …}`

List of pin/value pairs that will disable scan mode

`-scan_scale_power_factor <factor>`

Scale applied to power in scan dummy cell. Default: `number_of_total_pins / number_of_non-scan_pins`

`-si_immunity <name>`

Name of template for signal integrity immunity tables. The defined template is used for characterization of noise immunity rejection curves. The range of input noise widths and output loads to use are defined by the given template name where the template is pre-defined using the `define_template` command. The `-si` option of `the char_library` command must also be set for signal integrity characterization.

`-type <normal | mega | io>`

The type of cell. Default: `mega`

Note: In the `define_cell` command, the `-type uda` option has been renamed to `io`. Both `io` and `uda` are accepted and execute the same code. This means that they are synonymous.

Liberate uses a unique algorithm for automatic vector generation of larger (`mega`) cells that involves a more detailed pre-characterization analysis than is required for typical cells (`normal`). Mega cells are cells with a large number of pins, a large number of transistors, or both. Liberate will use the normal cell algorithm by default. If the mega cell algorithm is chosen, a message will appear in the log file. The `-type` option can be used to override the built-in selection criteria. The default is to let Liberate automatically choose which algorithm to use when characterizing each cell. Using mega mode will typically reduce both preprocessing and characterization time for large cells.

Note: Using `io` disables the automatic vector generation (the *Inside View* algorithm) of Liberate and characterizes only the user-defined arcs (see define_arc).

`-user_arcs_only` Skips the automatic addition of arcs by the *Inside View* algorithm of Liberate. This option has been provided to support the `char_library -user_arcs_only` functionality for a specific cell list. It affects only the cells listed in the `define_cell` command. You must provide all required arcs by using the `define_arc` command.

Note: This option is often used with the `write_template -verbose` command to ensure that the new library matches the reference library structure.

`-when <"function">` Specifies user-defined cell-level logic constraints using the Liberty format *when* syntax, constraining the automatic vector generation of Liberate for the given cell. The `define_cell -when` logical condition applies only to steady state signals such as leakage states and side input states that are non-switching. Liberate does not automatically infer simultaneous switching inputs based on the logical condition. You can use `define_arc` to specify simultaneous switching inputs, or specify a truth table to be translated automatically.

`{cell_names}` List of cell names to be characterized.

The -input, -bidi, -output, -clock, and -async options define the pin type for the given list of pin names. 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 will be ignored. For example, combinatorial cells such as NOR or NAND gates might not have clock or async pins. Therefore, any definition for these pins will be ignored. Likewise, if a pin name is specified but not used by a particular cell it will be 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, they must be defined uniquely with separate define_cell commands.

The -scan_* options are required when a dummy cell (one with the scan pins removed) is required. The -scan option specifies the names of the scan-related pins that are to be removed. The -scan_disable option provides the state information that will disable scan mode. If the application of the -scan_disable constants results in 2 or more data groups with the same states, then the group data will be merged according to the rules specified by the default_leakage, default_power, and default_timing parameters. (If the default group is turned off, then the group data will be merged using the default setting of *max*) The -scan_cell_prefix and -scan_cell_postfix options provide a string to attach to the beginning or end of the cell name when the scan pins are removed. If either of these options is specified, the library will contain both the original cell and the scan dummy cell with the modified name. When the write_library -scan_dummy_scale_power option is enabled, the power data in the scan dummy will be scaled according to the factor specified by the -scan_scale_power_factor option. This option specifies a scale to be applied to the remaining power tables when the scan pins are removed. The default power scaling factor is computed as the number of total pins divided by the number of non-scan pins.

This command must be used before **char_library**.

Examples

```
define_cell -input {A1 A2} -output {Z} \
-delay delay_3x3 -power power_3x3 \
{NAND2X4 NOR2X2}
define_cell \
    -input {D}\
    -output {Q QN}\
    -clock {CK}\
    -async {SN} \
-delay delay_5x5 \
-power power_5x5 \
-constraint constraint_3x3 \
{DFFX1}
define_cell \
    -input {A1 A2 A3 A4 SLP}\
```

```
    -output {Y}\
    -pinlist {A1 A2 A3 A4 SLP Y} \
-delay delay_5x5 \
-power power_5x5 \
-when "!SLP" \
{MTAND2 MTAND3 MTAND4}
# The following example attaches a 100 Ohm resistor between
# the PAD and PADN pins in addition to the load caps
# (from index_2) added to PAD and PADN.
======template.tcl===
define_cell \
    -input {I} -output {PAD PADN} \
    -pinlist {I PAD PADN} \
    -harness "Obuff_load" \
    -delay delay_template_OL \
    -power power_template_OL \
    OBUFF
======char.tcl=======
lappend spicefiles /home/work/custom_load/testload
======testload=======
.subckt Obuff_load PAD_altos_out PADN_altos_out
C1 PADN_altos_out 0 'PADN_cap'
C2 PAD_altos_out  0 'PAD_cap'
R1 PAD_altos_out PADN_altos_out 100
.ends

# Here is an example of a MUX testcase with an
# un-buffered input and one-hot data selectors that use
# the define_cell -harness capabilities.
======template.tcl===
define_cell \
    -input { S0 S0_B S1 S1_B I0 I1 } \
    -output { X } \
    -delay delay_template_3x3 \
    -power power_template_3x3 \
    -when "((S0 !S1) + (!S0 S1)) & \
        (S0 ^ S0_B) & \
        (S1 ^ S1_B)" \
    -pinlist {I0 I1 S0 S0_B S1 S1_B N X} \
    -harness mux_harness \
    { MUXI21 }
```

```
======char.tcl=======
lappend spicefiles mux_harness.spi
======harness subckt=======
.subckt mux_harness
+ I0_altos_in I0_altos_stim I1_altos_in
+ I1_altos_stim X_altos_out S0_altos_in
+ S0_altos_stim S0_B_altos_in
.inc 'INV_1.spx'
* driver for pin I0,
* I0_altos_stim -> I0_altos_tmp -> I0_altos_in
Xdriver0_1 I0_altos_tmp I0_altos_stim vdd vss INV_1
Xdriver0_2 I0_altos_in I0_altos_tmp vdd vss  INV_1
* driver for pin I1,
* I1_altos_stim -> I1_altos_tmp -> I1_altos_in
Xdriver1_1 I1_altos_tmp I1_altos_stim vdd vss INV_1
Xdriver1_2 I1_altos_in I1_altos_tmp vdd vss  INV_1
* driver for pin S0 and S0_B,
* S0_altos_stim -> S0_B_altos_in -> S0_altos_in
Xdrivers01 S0_B_altos_in S0_altos_stim vdd vss INV_1
Xdrivers02 S0_altos_in S0_B_altos_in vdd vss INV_1
* VDD_voltage is defined as a parameter by
* Liberate, <supply_net>_voltage
* .global is not supported to avoid net name
* conflicts with the cell being characterized
* and messing up power/leakage measurements
Vdd vdd 0 VDD_voltage
Vss vss 0 VSS_voltage
* X_cap is defined as a parameter by Liberate:
*     <port_name>_cap
* Here we are defining a PI load based on the
* index_2 value.
.param c_near=X_cap/2
.param c_far=X_cap/2
.param rshield=5
Cnear X_altos_out 0 c_near
Rsh  X_altos_out far_end rshield
Cfar far_end 0 c_far

.ends
```

define_cell_leakage

Creates the `define_leakage` states. This command supports the complete iterative expansion of a list of pins while holding a separate list of pins at a static (DC) level.

Use this command to expand a list of pins into separate `define leakage` commands. This command supports the complete iterative expansion of a list of pins in the `active_pinlist` while holding a separate list of pins specified in the static_pinlist at the values specified in the `static_vector`. In addition, different prevector sequences can be provided using the `prevector_pinlist` and `prevector_list`. The `prevector_when_list` associates specific prevector sequences from the `prevector_list` with a specific "`when`".

Options

`-echo` Echo each generated `define_leakage` command into the log file.

`-active_pinlist {`*`list`*`}`

List of pin names to expand into different states.

`-leakage_option <`*`string`*`>`

Specifies arbitrary legal `define_leakage` syntax to include in the generated `define_leakage` commands.

`-prevector_list {`*`list_of_values`*`}`

List of prevectors. For example, {"0 1 0" "1 0 1"}

`-prevector_pinlist {`*`list`*`}`

List of pins corresponding to `-prevector`.

`-prevector_slew`

Specifies the slew rate to apply to the input and bidi pins in the `-prevector_pinlist`. Supported values are: `<`*`value`*`>`, `min`, `mid`, `max`, *`index_n`*. For more information, see prevector_slew.

`-prevector_when_list {`*`list`*`}`

List of 'when' conditions corresponding to each prevector. For example, {"CK" "!CK"}

`-static_pinlist {`*`list`*`}`

List of pins that remain static (do not toggle).

`-static_vector {`*`list`*`}`

Vector string containing a value for each pin in the static pinlist.

`{cell_names}` Required list of cell names to which the `define_leakage` commands are applied.

This command must be specified after the define_cell command for the cells listed in the `define_cell_leakage` command and before the char_library command.

Examples

Example1

Generate `define_leakage` commands by expanding active pin “b” while holding pin “a” as static.

```
define_cell -input { a b } -output y -pinlist {a b y} na2
define_cell_leakage \
    -echo \
    -active_pinlist "b"
    -static_pinlist "a" \
    -static_vector "1" \
    na2
```

The following commands will be echoed into the log file and executed:

```
define_leakage  -when "a*!b" -vector "10x" na2
define_leakage  -when "a*b" -vector "11x" na2
```

Example 2

Generate `define_leakage` commands for two active pins and one static pin with prevectors associated with the state of `clk`.

```
define_cell -input { d en } -clock { clk } -output { q } -pinlist { d clk en q }  dff
define_cell_leakage \
    -echo \
    -prevector_pinlist {clk} \
    -prevector_list    {"0 1 0" "1 0 1"} \
    -prevector_when_list { "!clk"    "clk" } \
    -active_pinlist "d clk" \
    -static_pinlist "en" \
    -static_vector "1" \
    { dff }
```

The following commands will be echoed into the log file and executed:

```
define_leakage  -when "en*!d*!clk" -prevector_pinlist "clk" -prevector "0 1 0" -
vector "001x" dff
define_leakage  -when "en*!d*clk" -prevector_pinlist "clk" -prevector "1 0 1" -
vector "011x" dff
define_leakage  -when "en*d*!clk" -prevector_pinlist "clk" -prevector "0 1 0" -
vector "101x" dff
define_leakage  -when "en*d*clk" -prevector_pinlist "clk" -prevector "1 0 1" -
vector "111x" dff
```

Example 3

```
# Call the following command prior to characterization.
define_cell_leakage -echo \
    -static_pinlist "A B C D E" \
    -static_vector  "0 0 1 1 1" \
    -active_pinlist "F G H I J" \
    -leakage_option "-ignore_for_default" \
    myCell
```

define_duplicate_cell

Specifies a list of cells that will be duplicated using the characterization data. The characterization data must exist for the master cell. The duplicated cells are not characterized. If the `-cells` option of the `write_library` command is used, the duplicated cells must be specified with this option to be written to the output library.

Options

<cell>	The name of the master cell to be duplicated.
{ cellname }	List of cell names duplicated from the master.

This command can be specified before `write_library`. If specified before `char_library`, the duplicated cells are not characterized.

Examples:

```
define_duplicate_cell INVX1 {INVX2 INVX3}
define_duplicate_cell INVX1 {INVX4}
```

define_duplicate_pins

Specifies a list of pins for a cell that will not be directly characterized, but instead will be given duplicate data from a characterized pin. This command copies all the pin data from the *<pin>* to each of the duplicated pins including any data where the *<pin>* is a related_pin. If the pin is an input pin, the duplicate input pin will include pin cap, hidden power, constraints, and CCSN (CCS noise). In addition, all input to output arcs where the pin is a related_pin will be duplicated for each `duplicate_pin`. For example:

```
define_duplicate_pin mux A { B C D E F G H I }
```

Options

<cellname>	The cell name to apply the duplication to.
<pin>	The pin to be duplicated.
{ *duplicate_pins* }	List of pin names to be duplicated.

This command also supports duplicating a complete bus. For example:

```
define_duplicate_pins myCell addrA addrB
```

...where `addrA` and `addrB` are buses. The following restrictions apply:

- `addrA` and `addrB` must have the same "direction" (cannot duplicate an input to an output.)
- `addrB` cannot be of a lower range than `addrA`, but `addrB` can be of a larger range than `addrA`. In this case, only the 1st *n* bits of `addrB` will be duplicates of `A` and the rest will remain unique. *(In other words, if addrA is 16 bits, and addrB is 8 bits, you can clone addrA[7:0] to addB[7:0]. But if addrA is 8 bits, and addrB is 16 bits, you cannot clone anything into addrB[15:8] from addA, because addA doesn't contain that bit range.)*
- `addrB` does not need to exist - it can be created automatically during `write_library`.
- Duplicating bundles is not supported.

define_group

Defines a cell group and a text description for that group. This information is used by the datasheet generator (write_datasheet), `compare_library -group`, `compare_structure` (only available in Liberate_LV), and `write_template` to group cells with similar functionality. The `define_group` command must occur after a **char_library**, **read_ldb**, or read_library command. To save the group definitions in the library database (ldb) a subsequent write_ldb command must be used. If `define_group` is not used, cells are grouped based on their footprint. If no footprint information is available then all cells belong to their own unique group.

The grouping information is accessible through two API functions, `ALAPI_cellgroups` and `ALAPI_description`. The API function `ALAPI_cellgroups` generates a list of groups from the library, while `ALAPI_description` returns the text description of a given cell. For more information on these API functions, see the *Liberate API Reference Manual*.

Options

< group >	Group name.
<"description">	Text describing the group.
{ cell_names }	List of cells belonging to this group.

Example

```
# Define a group of OR gates
define_group OR2_gates "2 input OR" {OR2_1 OR2_2 OR2_4}
```

define_index

Overrides the indexes specified in the templates referenced by define_cell, or created using the -auto_index option of char_library, for all the arcs between the -related_pin list and the -pin list for the given type. Valid table types are: constraint, delay, mpw, power, and si_immunity. The overrides apply only to the cells listed in the define_index command.

Note: If a define_index constraint that does not refer to any arc exists, the tool attempts to apply it with -pin and -related_pin reversed. For this reason, it is desirable to remove any define_index constraints if the corresponding define_arc is removed to avoid having it applied incorrectly.

Options

`-index_1 {`*`indexes`*`}` List indexes to use as index_1.

`-index_2 {`*`indexes`*`}` List indexes to use as index_2.

`-pin {`*`pins`*`}` List of pin names (REQUIRED).

Note: The `-pin` option accepts the asterisk "`*`" wildcard character. For example:

```
define_index -pin D*  # All pins beginning with "D"
define_index -pin *  # All pins
```

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

List of related pin names. Accepts wildcards.

The `-related_pin` option is required for most arcs, but is not required for arcs that need only one pin, such as hidden power arcs and mpw or `min_period` arcs.

Note: The `-related_pin` option accepts the asterisk "`*`" wildcard character.

`-type {data_types}` List of data types *constraint,* `ccsn_dc`, `delay`, `mpw`, `power`, or `si_immunity`.

The `ccsn_dc` template type is used for composite current source DC noise model characterization. For more information on the `ccsn_dc` template, see the `define_template` command.

`-when <"`*`function`*`">` Logic state of side inputs.

`{`*`cell_names`*`}` List of cell names. This option accepts wildcard characters.

Important Points to Note

- For variation characterization in Liberate Variety or in unified characterization flow of Liberate Trio, the minimum dimension of a template is 2x2.
- The `-pin`, `-related_pin`, and at least one of `-index_1` or `-index_2` options must be specified.
- The size of the `-index_1` and the `-index_2` lists must be equal to the equivalent template type specified by `define_cell`. The `-when` option provides for defining unique indexes using the Liberty when syntax.

- Multiple `define_index` commands can be used to specify different overrides for different arcs for the same set of cells, or for different cells.
- The `define_index` command must follow the `define_cell` command and it must precede the `char_library` command.

  Note: If with the `char_library` command you used the `-auto_index` option, the `define_index` command is applied *after auto_index* completes. If `define_index` is successful (correctly specified with cell/pin etc), it will override the index values determined by `-auto_index`.

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 {0.010 0.050 0.500} \
    {NAND2X4 NOR2X2}
```

define_input_waveform

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

Important

All input transition values must be specified as PWL for a given index of slews. It is not valid to specify one index value as PWL and have the others default to another alternative input waveform method.

Options

`-direction < rise | fall >`

Specifies a logical direction of the given input waveform. The valid direction values are `rise` and `fall`. (REQUIRED)

`-gnd_val <voltage>` Specifies the input gnd voltage value of the associated input pin(s) in volts. (REQUIRED)

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

Specifies a list of cell-pin pairs to apply the waveforms. The cell and pins can be specified using wildcards. Examples are: Wildcards (*) are supported in place of explicit cell/pin names. User Inputs for a cell/pin is searched for in the following order: `cell:pin`, `cell:*`, `*:pin`, and `*:*`. If Normalized Driver Waveform (NDW) data needs to be written in the output library, use of wildcards is not recommended in the pin names. For more information about use of wildcards and NDW, see driver_waveform_wildcard_mode.

`-pwl {time voltage <time voltage> …}`

List of paired values of time and voltage in MKS units. (REQUIRED)

`-scale` Scale the normalized voltages in the PWL by vdd. *(Reserved for write_template flow - not a user option.)*

`-slew_index <slew_index>`

Specifies the index transition value that the pwl waveform is linked with. The slew from the index is specified in LDB or .lib units. (REQUIRED)

`-vdd_val <voltage>` Specifies the input vdd voltage value of the associated input pin(s) in volts. (REQUIRED)

While characterizing NLDM delay models, you can use this command for defining the input waveform shape. For detailed information, see Delay Models in Chapter 7, “Performing Characterization using Liberate.”

This command must be used before char_library.

Example

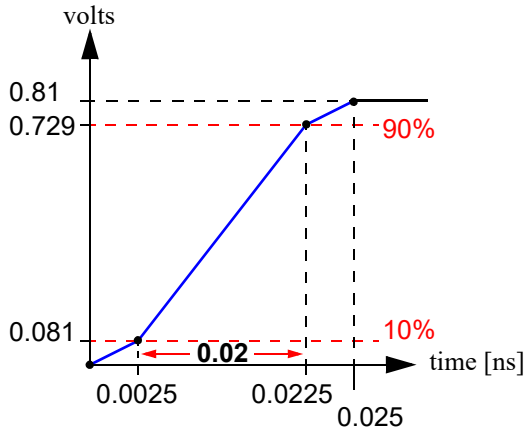
This sample shows a template file that has slew thresholds set at 10% and 90%, and slew index values set at 0.02 and 0.555:

```
set_var slew_lower_rise 0.1
set_var slew_upper_rise 0.9
set_var slew_lower_fall 0.1
set_var slew_upper_fall 0.9
set_var measure_slew_lower_rise 0.1
set_var measure_slew_lower_fall 0.1
set_var measure_slew_upper_rise 0.9
set_var measure_slew_upper_fall 0.9



define_template -type delay \
    -index_1 {0.02  0.555} \
    -index_2 {0.08  0.71 } \
    delay_template_2x2
```

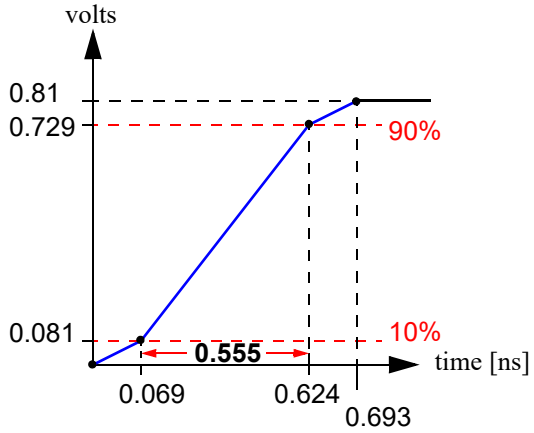
Liberate checks that the PWL waveform supplied matches the `slew_index`. In this example, the PWL waveform must be defined so the slew time measured from the `measure_slew_lower_rise` to the `measure_slew_upper_rise` thresholds matches the `slew_index` value. If the `measure_slew_*` and the `slew_*` parameters are not the same, then the slew time derived from the waveform must be derated before it is checked against the `slew_index`. The derate factor is computed as follows:

`slew_derate = (measure_slew_upper_rise - measure_slew_lower_rise)/(slew_upper_rise - slew_lower_rise)."`

Both rising and falling waveforms must be defined. *(Only rising waveforms shown in this example.)*



Rising waveform
Slew = 0.02



Rising waveform
Slew = 0.555

```
# Definitions for rising PWL waveforms. Falling waveforms are similar.
define_input_waveform \
    -direction rise \
    -pwl {0.0 0.0  0.0025e-9 0.081  0.0225e-9 0.729  0.025e-9 0.81} \
    -vdd_val 0.81 \
    -gnd_val 0.0 \
    -slew_index 0.02 \
    -pinlist {INV_X1 A}

define_input_waveform \
    -direction rise \
    -pwl {0.0 0.0  0.069e-9 0.081  0.624e-9 0.729  0.693e-9 0.81} \
    -vdd_val 0.81 \
    -gnd_val 0.0 \
    -slew_index 0.555 \
    -pinlist {INV_X1 A}
```

define_leafcell

Specifies the bottom level (leaf) of the hierarchy in a cell level netlist. It is the boundary between the cell netlist and the model file. It also specifies the device type of the leaf devices so that the *Inside View* algorithm can correctly identify the circuit functionality. This allows Liberate to correctly identify devices in the cell netlist even when the process model file cannot be parsed by Liberate but can be parsed by the external circuit simulator. This command can be used in combination with the `extsim_model_include` parameter to enable external simulation (see `-extsim` option of the `char_library` command) with the process models and the compiled netlist. This command supports identification of mosfets, diodes, resistors, capacitors, and other device types.

Options

`-area "`*`parameter_name`*`"`

Specifies the name of the diode area parameter in the cell. Default: "`area`"

`-element`

Enables circuit element-based leaf cells. Model name(s) must be specified.

Without this option, the `define_leafcell` command applies only to instances in the netlist. An instance in SPICE format has a name beginning with "X". When this option is enabled, the `define_leafcell` command is applied only to the circuit elements (such as R, C, M) instead of instances. This option supports circuit elements with a model, and a pin count greater than 2 (for example, 3-terminal R's and C's). Use this option when the netlist contains elements such as "M" (Mosfet), "D" (Diode), and so on.

Note: If the netlist has instances (preceding "X"), this option should not be used.

`-extsim_model` Allows for partial `include` and partial `read_spice`.

The `-extsim_model` option loads the model files for the leaf cell(s). If this option is used, the leaf cell being defined also needs to have `extsim_deck_header` insert a "`.inc'<`*`path`*`>/modelfile.inc'`" to load a model for this cell – most likely a Verilog model. If one or more leaf cells does not have the `-extsim_model` option nor the extsim_model_include parameter present, the tool will output an error requesting use of the `extsim_model_include` parameter and quits.

`-length "`*`parameter_name`*`"`

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

`-multiple "`*`parameter_name`*`"`

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

`-nfin "`*`parameter_name`*`"`

Specifies the name of the nfin parameter in the netlist instance (or element) that maps to the number of fingers of a FinFet process. Default: `'NFIN'`

`-pin_position {`*`list_of_pin_positions`*`}`

Maps the pins of the leaf cell in the netlist to the pins of the corresponding subcircuit or model in the model file. There should be only one number for each pin in the leaf cell. The first pin is designated by `0`. (REQUIRED)

Pin positions usually start from `0 <drain gate source [bulk(s)]> | <terminal_p terminal_n [bulks]>`

`-pj "`*`parameter_name`*`"`

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

`-scale "`*`value`*`"` Specifies the mos parameter scale factor in the cell. This scale factor is used only by the *Inside View* algorithm of Liberate to determine device sizes, and is not applied to the device sizes in the simulation netlist. Default: `'1.0'`

`-type < nmos | nmos_soi | pmos | diode | r | c | nmos_stk | pmos_stk | pmos_soi | npn | pnp | black_box >`

Type of cell. For related details, see Using the -type option.

`-width "parameter_name"`

Specifies the name of the mos Width parameter in the cell. Default: `'w'`

`{leaf_cell_names}` List of leaf cell names

This command must be used before `read_spice`. If the `-extsim_model` option is used with this command, it must be used before char_library and write_library as well.

If the `define_leafcell` commands in the characterization script have the `-extsim_model` option included, you can load the model files for these leaf cells by using the extsim_model_include or extsim_deck_header parameter. All other device models can be loaded by using `read_spice`.

Using the -type option

The `-type` option specifies the type of the cell. Supported settings are `nmos`, `pmos`, `diode`, `r`, `c`, `nmos_stk`, `pmos_stk`, `npn`, `pnp`, and `black_box`. The `nmos_stk` and `pmos_stk` types support five pin stacked NMOS/PMOS transistors. For seven pin stacked MOS, the extra two pins are internal pins. The `pin_position` for stacked MOS is `'d g1 g2 s b'`.

The `npn`, and `pnp` types identify circuit instances that are Bipolar npn or pnp transistors. These types support a `pin_position` with 3 positions (0, 1, and 2).

The `black_box` instance type supports as many pins in -pin_position as are required. The Liberate internal simulator, Alspice does not support BJT devices or black boxes. If the circuit has any of these devices, then an external simulator must be used with `-io` (see char_library `-extsim -io`). In addition, `extsim_model_include`, `define_leafcell`, `extsim_flatten_netlist` (`=0`) and `define_arc` must be used.

Note: Each `type` requires a minimum number of pins/element in the `pin_position` option.

The `nmos_soi` and `pmos_soi` types are used to identify Silicon on Insulator (SOI) leaf cell mosfets. The supported `pinout` of the `nmos_soi` and `pmos_soi` related types is:

```
MXX d g s e [p] [b] [t] mname [L=val] [W=val] ...
```

where:

```
MXX is the instance name.
d is the drain
```

```
g is the gate
s is the source
e is the back gate (or substrate),
p is the external bulk
b is the bulk
t is the temperature node.
mname is the model name.
```

The drain, gate, source and back gate nodes are required. Up to 7 terminal nodes are supported.

Examples

Example 1

```
# Define the cell NCH_MAC as a leaf cell
define_leafcell \
        -type pmos \
        -pin_position { 0 1 2 3 } \
        PCH_MAC
# Define the cell PCH_map as a leaf cell.
# 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
```

Example 2

```
set_var extsim_deck_header ".hdl /support/diode.va"
define_leafcell -extsim_model -type diode\
    -pin_position {0 1} {diodeva}
set spicefiles "netlist.sp"
lappend spicefiles "/support/sp_models.inc"
# Read in spectre netlists
read_spice -format spectre $spicefiles
```

Example 3

```
# Support 3-terminal transistors (PODE) with define_leafcell -pin_position to
# map a leafcell terminal to multiple MOS ports (d/g/s/b). If the PODE (S=D)
# is a 3-terminal subckt with pins (D G B), it can be defined as:
```

```
define_leafcell -type nmos -pin_position {0 1 0 2} {npode_mac}
```

define_leakage

Defines the logic conditions to use for calculating leakage power for the given cells. Using this command, you can disable the automatic determination of leakage conditions by Liberate for the listed cells.

Options

`-extsim_deck_header`

Allows to provide external simulator commands directly to the external simulator on an individual arc basis without having Liberate 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_library command). It is a local arc specific version of the `extsim_deck_header` parameter. 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_leakage -extsim_deck_header ".ic n128 0" \
    -related_pin ck -pin Q ..
```

`-ignore_for_default`

Instructs that the specified leakage should not be considered in any calculations when computing default `leakage_power` groups or attributes. (See set_default_group)

For example:

```
...
read_ldb My.ldb.gz
define_leakage -when "a*!b" -ignore_for_default MyCell
write_library My.lib
```

Note: This option must be set in a `define_leakage` command before the write_library command.

`-load_dir <U | D | B>`

Output load direction (pullUp, pullDown or Both).

`-no_harness` Instructs to not include the harness in the simulation deck for the specified leakage state if the cell has a harness specified (see define_cell `-harness`).

`-pinlist { list_of_pins }`

List of pins corresponding to vector.

`-pin_load <template>`

Predefined pin loading. (See define_pin_load)

`-prevector <"vector">`

User-specified initialization vectors. Default: no prevectors

`-prevector_pinlist { list }`

User-specified pin list for pre-vectors. For related details, see Using the -prevector option.

`-when <"function">` Specifies the logic conditions using the Liberty *when* format syntax. (REQUIRED)

This option should be used when characterizing I/O cells that do not use automatic vector generation (`char_library -io`).

Enhanced leakage characterization is also achievable by the tool honoring the `-when` condition specified in the define_cell command.

`-value { supply value }`

Specifies a list of pairs of supply name and leakage value. Default: use the characterized leakage value

Example:

```
define_leakage -value { \
    VDD $vdd_leak_val VSS $vss_leak_val } \
    $cellname
```

`-vector <"stimulus">`

Vector stimulus used to simulate the leakage, where each bit can be one of the following: `X`, `1`, or `0`.

The `-vector` option allows for partial `when` conditions for leakage in the output library, while having secondary pins set to a mix of 0s and 1s.

`{ cell_names }` List of cell names.

This command must be used before `read_spice`, char_library, and write_library.

Examples

Example 1

```
# Define the leakage conditions of an IO cell
define_leakage -when " D * !EN"        IO_cell
define_leakage -when "!D * !EN"        IO_cell
define_leakage -when " D *  EN *  PAD" IO_cell
define_leakage -when " D *  EN * !PAD" IO_cell
define_leakage -when "!D *  EN *  PAD" IO_cell
define_leakage -when "!D *  EN * !PAD" IO_cell
```

Example 2

```
# Define the leakage conditions for cell, MYCELL, including the simulation stimulus
# to put the device in a user-defined state before proceeding with characterization
define_leakage \
    -pinlist  { D CK Q } \
    -prevector_pinlist {D CK} \
    -prevector "00 01 00 10" \
    -vector "100" \
    -related_pin CK \
    -pin Q \
    MYCELL
```

define_map

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 cells name(s) returned by the API functions: `ALAPI_inputs`, `ALAPI_outputs`, `ALAPI_inouts`, `ALAPI_internals`, `ALAPI_clocks`, `ALAPI_pinnames`, `ALAPI_name`, `ALAPI_cellnames`, and `ALAPI_cellgroups`.

Options

<map_filename> Name of the map file

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, it must be used before the read_ldb and **read_library** commands.

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_file 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_attr_limit

Sets a pin-specific maximum capacitance attribute limit.

Options

`-cells {list}`	List of cells.
`-pinlist {list}`	List of pins.
`< limit >`	The `max_capacitance` attribute limit value (in Farads).

The values set by `define_max_capacitance_attr_limit` override the calculated `max_capacitance` attribute when the limit is exceeded. Multiple `define_max_capacitance_attr_limit` commands can be specified. A wildcard `*` is supported to allow the cell name to reference all cells. Only cells with the given pin name(s) will be effected. A wildcard cannot be used for the pin.

This command must be used before model generation (write_library).

Example

```
# Set the max capacitance for pin Y of the cell AND1
define_max_capacitance_attr_limit \
    -cell AND -pinlist Y 100e-15
```

define_max_capacitance_limit

Sets a pin-specific maximum capacitance.

Options

`<value>`	Maximum allowable capacitance (in Farads).
`{<cell> <pin> ...}`	List of cell pin pairs.

This command has effect only when you use the `-auto_index` option with the char_library command. 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. The wildcard character `*` is supported to allow the cell name to reference all cells. Only cells with the given pin name will be effected. A wildcard cannot be used for the pin.

This command must be used before `char_library`.

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 `max_transition` value used by the `auto_index` algorithm.

Options

`<value>`	The `max_transition` value (in seconds).
`{<cell> <pin> ...}`	List of cell-pin pairs. The wildcard asterisk (`*`) is supported to allow the cell name to reference all cells. A wildcard cannot be used for the pin. Only cells with the given pin name will be effected.

This command has effect only when using the `-auto_index` option with the write_library command. The values set by `define_max_transition` override the global value set by the max_transition parameter and are used by the `auto_index` algorithm when it computes the maximum load index. Multiple `define_max_transition` commands can be specified.

If this command is used with the VDB flow, care should be taken to not apply double-scaling. For example, the write_vdb command should not use -auto_index. Instead, your characterization script (that is, char.tcl) should have a read_vdb -> char_library -auto_index flow. If both scripts use -auto_index, the resulting library will have max_transition scaling applied twice.

This command must be used before `char_library`.

Example

```
# Set the default maximum transition time
set_var max_transition 1e-9
# Set maximum transition time for some clock pins
define_max_transition 0.75e-9 {DFFX1 CK LTX1 LCK}
define_max_transition 0.5e-9 {GATER CLKIN * CLK}
char_library -auto_index
```

define_min_transition

Sets a pin-specific minimum transition.

Options

`<value>`	Minimum allowable transition time (in seconds).
`{<cell> <pin> ...}`	List of cell/pin pairs.

This command has effect only when you use the `-auto_index` option with the char_library command. The values set by `define_min_transition` override the global value set by the `min_transition` parameter. Multiple `define_min_transition` commands can be specified. A wildcard `*` is supported to allow the cell name to reference all cells. Only cells with the given pin name will be effected. A wildcard cannot be used for the pin.

If this command is used with the VDB flow, care should be taken to not apply double-scaling. For example, the write_vdb command should not use -auto_index. Instead, your characterization script (that is, char.tcl) should have a read_vdb -> char_library -auto_index flow. If both scripts use -auto_index then the resulting library will have `min_transition` ***scaling applied twice.***

This command must be used before `char_library`.

define_out_to_out_arc

Instructs Liberate to characterize an arc from an output pin to an output pin. This command will merge a characterized arc A->X with arc A->Y into arc X->Y.

Options

`-related <pin>`	The related_pin for both forward arcs. This option specifies the name of the input pin whose transition triggers the arc.
`-related_out <pin>`	The intermediate transitioning output. This option specifies the first output to switch and becomes the `related_pin` for the out-to-out arc.
`-out <pin>`	The last transitioning output. This option specifies the output pin for both forward arcs and is the pin in the output library.
`-cells {list_of_cells}`	List of cells.

This command must be used before write_library.

Example

```
define_out_to_out_arc -related A -related_out X -out Y -cells { mycell }
```

define_pin_load

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

Options

`-pullup_voltage <value>`

Pullup voltage.

`-pulldown_voltage <value>`

Pulldown voltage.

`-pullup_resistance <value>`

Pullup load resistance in ohms.

`-pulldown_resistance <value>`

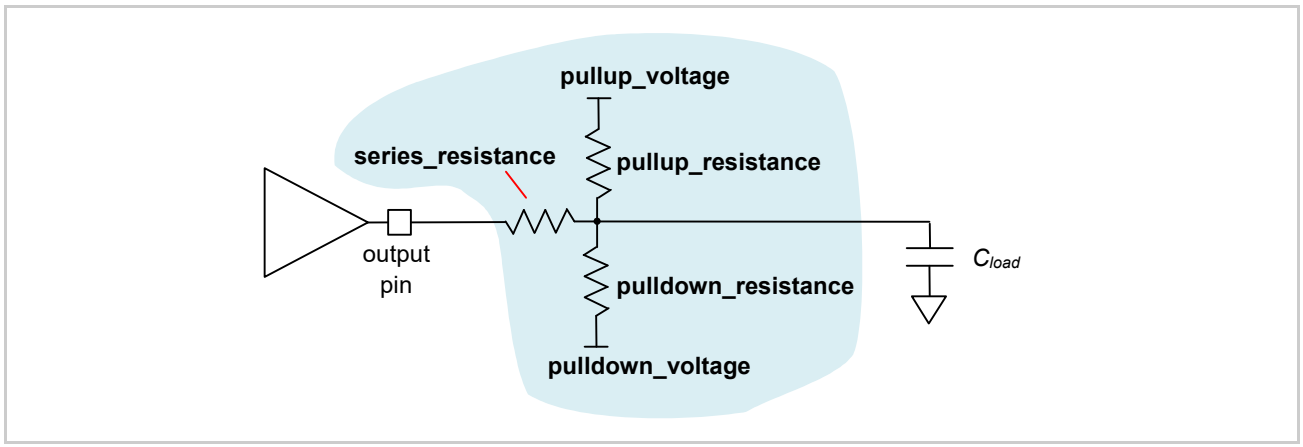
Pulldown load resistance in ohms.

`-series_resistance <value>`

Series load resistance in ohms.

`<pin_load_name>` Name of pin load definition. (REQUIRED)

The loading of the pin can consist of a pullup voltage source via a pullup resistance, a series resistance and a pulldown resistance tied to a pulldown voltage source.Resistances must be specified in ohms.



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.

This command must be used before char_library.

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_pulse_generator_arc

Identifies arcs that generate a pulse on an output pin.

Note: If advanced CCS or ECSM format models are required, then IO mode, which is enabled using the char_library `-io` command, may be required with a verbose template.

Options

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

Specify a required list of cells. The cell name accepts wildcard. Default: `""` (none specified)

`-pin <`*`pin_name`*`>` Specify the pin name (required). Default: `""` (none specified)

`-related_pin <`*`pin_name`*`>}`

Specify the related pin name (required). Default: `""` (none specified)

`-when <"`*`function`*`">` Specify the logic condition of the side inputs. Default: `""` (none specified)

This command can be executed multiple times, once for each pulse generator arc. For the specified arcs Liberate will subtract `CV^2` from the fall power. Then, divide the fall power by two and copy the fall power into the rise power. This is done because a single transition on the input generates two transitions on the output, one for the falling arc power and one for the rising arc power and we do not want the downstream power tool to double count the power.

If the `-when` option is not specified, this command is applied to all arcs that match the specified pin and related_pin. If `-when` is specified, this command is applied only to those arcs that match the specified `when` condition. An exact match occurs when all pins are present and are in the same state. The order of the pins is not considered.

This command must be used before char_library.

Example

```
define_cell \
    -clock { CK } \
    -input { EN } \
    -output { CKO } \
    -pinlist { CK EN CKO } \
    -delay delay_template \
```

```
    -power power_template \
    -constraint constraint_template \
    PULSE_GEN
define_pulse_generator_arc -cells PULSE_GEN -pin CKO -related_pin CK -when "EN"
```

define_pvt

Defines a PVT by name and all the settings that are associated with the specific PVT. The name specified must be unique.

Note: All PVTs must be defined before they are referenced by the `set_pvt` command.

For more information, see Chapter 9, "Using the Multi-PVT Characterization Flow of Liberate Trio."

See also get_pvts and set_pvt.

Options

`-dont_overwrite`	Specifies not to overwrite the existing PVT definition. Default: `false`
`-default`	Specifies to use the given PVT definition as the default PVT for generating all the `when` conditions and vectors to drive the multi-PVT characterization. Default: `false` This option also forces execution of all the Tcl `<commands>` passed to the `define_pvt` command immediately when read by the Liberate server process. The default option is ignored when the `define_pvt` command executes on a slave or Liberate client process.
`<pvtName>`	Specifies the name of the PVT corner.
`<commands>`	Lists the commands that are specific to the given PVT corner.

Example

```
# Define the default corner
define_pvt -default typical_max_100c {
    set pvt typical_max_100c
    set VDD 0.54
    set temp 100
}
# Define all additional corners
define_pvt cbest_min_130c {
    set pvt cbest_min_130c
    set VDD 0.65
```

```
    set temp 130
}
define_pvt cworst_max_0c {
    set pvt cbest_max_0c
    set VDD 0.54
    set temp 0
}
…
```

define_template

Defines a template to be used for characterization.

Options

`-type {delay | power | ccs | ccsn_dc | constraint | ecsm | ecsm_vivo | mpw | si_iv_curve | si_immunity }`

Specifies the type of template being defined. (REQUIRED)

`-index_1 {`*`values`*`}` List values to be used as the first index. (REQUIRED)

Note: When you use the `ecsm_vivo` template type, this option should specify a list of normalized input voltage values.

`-index_2 {`*`values`*`}` List values to be used as the second index.

Note: When you use the `ecsm_vivo` template type, this option is ***required*** to specify a list of normalized output voltage values.

`-index_3 {`*`values`*`}` List values to be used as the third index.

`<`*`template`*`>` Name of template.

Note: For variation characterization in Liberate Variety or in unified characterization flow of Liberate Trio, the minimum dimension of a template is 2x2. Templates smaller than 2x2 can be defined and stored in a library, but cannot be used for variation characterization. In addition, ensure that all `-index_*` options for all the library constructs should be monotonically increasing.

This command must be used before char_library.

Using the -type option

The `-type` option specifies the type of 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.

Note: Internally, `define_template` uses a fixed set of units (listed below). These cannot be changed. *However*, units can be changed when writing a library with the set_units command.

current: 1mA (milliamps)

power: 1nW (nano watts)

resistance: 1kohm (kilohm)

time: 1ns (nano seconds)

voltage: 1V (volts)

capacitance: 1pf (pico farad)

The `delay` template type is 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.

The `power` template type is used for switching and hidden (internal) power 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.

The `ccs` template type can be used for composite current source model (CCS) delay characterization. It requires `index_1` to be specified where `index_1` represents the range of the normalized voltage values to measure.

The `ccsn_dc` template type can be used for Composite Current Source (CCS) DC noise model characterization. It requires `index_1` and `index_2` to be specified where they represent a range of input/output voltages. If the `ccsn_dc` template is not defined, Liberate defaults to a range of 29 voltage points (see ccsn_dc_template_size) from `-VDD` to `2*VDD`.

DC simulations are very fast, especially on a small CCB group of transistors extracted for noise stage simulations. Therefore, DC simulations do not use much CPU time compared to the transient CCS noise models. It usually is not necessary to change the size of these tables from the default of 29x29. The `ccsn_dc` template type can be useful to optimize the size of the tables to avoid non-convergence errors or because the process model itself is not stable over the default voltage range.

Ensure that specified `index_1` and `index_2` adhere to the following rules:

- The last entry should be greater than `1.01x VDD`.
- Index values must be monotonically increasing with no step exceeding 10% of the VSS to VDD voltage range.

The `ccsn_dc` template type is global to all cells. It is not referenced in the define_cell command like other templates such as `delay`, `power`, and `constraints`, and therefore cannot be specified on a per-cell basis.

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.).

The ecsm template type can be used for effective current source model (ECSM) characterization. It requires index_1 to be specified, where index_1 represents the range of the normalized voltage values to measure.

The ecsm_vivo template type can be used for ECSM VIVO characterization. It requires index_1 and index_2 to be specified, where:

- index_1 represents the range of the normalized input voltage values to measure, and
- index_2 represents the range of the normalized output voltage values to measure.

The mpw template type is used when a two dimensional mpw table is required. The user must provide both index_1 and index_2. In addition, the define_cell -mpw option must reference the mpw template. By default, if you do not provide an mpw template, the mpw timing constraint will be characterized as a single attribute. If the mpw_table parameter is set, Liberate will output a one dimensional mpw table using the list of values for the define_cell constraints. Use the define_template -type mpw command only when a two-dimensional MPW table is required.

The si_iv_curve template type is used for steady state I/V characterization. It requires index_1 to be specified, where index_1 defines the number of sample voltage points between supply and ground for steady state high and between ground and supply for steady state low. Each sample point is equally spaced within the total voltage range.

The si_immunity template type is used for noise immunity rejection curve characterization. It requires both index_1 and index_2 to be specified, where index_1 represents the range of input noise widths and index_2 represents the range of output loads.

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
# Power Template for 3 input slews, 3 output loads
define_template -type power \
-index_1 {0.025 0.1 0.25} \
-index_2 {0.0010 0.015 0.100} \
power_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

# ECSM  template for 5 intervals
define_template -type ecsm \
-index_1 {0.05 0.2 0.5 0.8 .95} \
ecsm_5

# si_iv_curve template for 11 intervals
define_template -type si_iv_curve \
-index_1 {0 1 2 3 4 5 6 7 8 9 10} \
Si_iv_11

# si_immunity template for 3 noise widths, 3 loads
define_template -type si_immunity \
-index_1 {0.100 0.50 3.00} \
-index_2 {0.0010 0.015 0.100} \
Si_immunity_3x3

# CCS Noise DC Curve with 11 input and 11 output voltages.
define_template -type ccsn_dc \
-index_1 {-1.0 -0.5 -0.2 -0.1 0.0 \
 0.1 0.2 0.5 1.0 1.2 1.5} \
-index_2 {-1.0 -0.5 -0.2 -0.1 0.0 \
 0.1 0.2 0.5 1.0 1.2 1.5} \
ccsn_dc_template
```

delete_arc

Deletes one or more arcs from a library.

Options

`-cell {cell_name}`	(Required) Specifies the cell name from which the arc is to be deleted.
`-pin "name"`	Specifies the pin of the arc to be deleted. (Required)
`-related "name"`	Specifies the related_pin of the arc to be deleted. Default: `"*"` (delete all arcs of the given type regardless of the related pin)
`-timing_sense "string"`	Specifies the 'timing_sense' of the arcs to be deleted. Default: `""` (delete all arcs of the given type regardless of the timing_sense)
`-timing_types {list}`	Specifes a list of the 'timing_type'(s) of the arcs to be deleted. Default: `{}` (delete all arcs of the given type regardless of 'timing_type')
`-type {list}`	Supported types are: `ccs`, `ccs_retain`, `constraint`, `delay`, `hold`, `mpw`, `power`, `recovery`, `removal`, `retain`, `setup`, `timing`. Default: delete all data types.
`-when "value"`	Specifies the when of arc to be deleted, Default: `"*"` (delete all arcs of the given type regardless of the when condition)

Use this command to specify the arcs to be deleted from the output library. This command does not affect characterization. It operates in the programming interface (ALAPI) level post characterization. Therefore, any function or command that uses ALAPI respects this command. Any function or command written in C language does not respect this command. Some commands, such as `write_ldb`, `write_socv` and `write_variation_table`, do not support `delete_arc`.

This command must be used before model creation.

Example

```
delete_arc \
    -pin {CEN} \
    -type {setup hold} \
    -cell  cell_name
    write_library -filename my.lib  myLib
```

distr_on_master

Returns `1` if called on a master in the arc packet flow, otherwise returns `0`. This command is useful for the commands that run only on the master.

Options

None

Example

```
if {[distr_on_master]} {
     puts "this is running on the master"
     # other commands running only on the master
}
```

distr_on_client

Returns `1` if called on a client in the arc packet flow, otherwise returns `0`. This command is useful for the commands that run only on the clients.

Options

None

Example

```
if {[distr_on_client]} {
     puts "this is running on the client"
     # other commands running only on the client
}
```

distr_on_client_assembly

Returns `1` if called on a client that is running an assembly job or a one-shot job in the arc packet flow, otherwise returns `0`. This command is useful for the commands that run only on the clients used to write out cell libraries.

Options

None

Example

In `char.tcl`:

```
if {[distr_on_client_assembly]} {
     puts "this is on a client that is running assembly jobs"
     # other commands running only on client running assembly jobs
}
```

distr_get_onecell_lib

Returns full path to the `.lib` file from the `write_library` command that was called just before this command. `distr_get_onecell_lib` is useful when used with the distr_on_client_assembly command to access a `cell.lib` written from a client when characterization is not finished for the other cells.

Note: This command must be used after the `write_library` command is successfully completed in the same session.

Options

None

Example

In `char.tcl`:

```
write_library …... output.lib
write_library …... output.lib_ccs_tn
if {[distr_on_client_assembly]} {
     set cell_ccs_lib [distr_get_onecell_lib]
     # running LDBX script on the ccs_tn lib of a cell and a pvt written out from
       a client.
     ldbx_python script.py $cell_ccs_lib
     # other commands running with $cell_ccs_lib as input
}
write_library …... output.lib_ecsm_ccs
```

distr_get_onecell_libs

Returns full path to all the `.lib` files from the `write_library` commands that were called just before this command. `distr_get_onecell_lib` is useful when used with the distr_on_client_assembly command to access all the `cell.lib` written from a client when characterization is not finished for the other cells.

Note: This command must be used after the `write_library` command is successfully completed in the same session.

Options

None

Example

In `char.tcl`:

```
write_library ….. output.lib
write_library ….. output.lib_ccs_tn
write_library ….. output.lib_ecsm_tn
if {[distr_on_client_assembly]} {
     set cell_libs [distr_get_onecell_libs]
    # running LDBX script on all libs of a cell and a pvt written out from a client.
     # in this case they are lib, lib_ccs_tn and lib_ecsm_lib three libs from this
cell and pvt
     ldbx_python script.py $cell_libs
     # other commands running with $cell_libs as input
}
```

distr_get_onetype_libs

Returns a list of full paths to `.libs` of groups of cells (all cells if the `-cells` option is omitted) according to the options specified with this command. `distr_get_onetype_libs` is useful when running on master to get a list of cell libraries generated from a post-process script after all the client jobs are finished, which will be later used in the input library list for `append_library`.

Options

`-cells <list_of_cell_names>`

[Optional] When specified, returns cell libraries for cells specified in the list. Otherwise, cell libraries for all cells are returned.

`-libtype < lib | lib_ccs_tnp | lib_ccs_tn | lib_ecsm_tn | lib_ecsm_ccs >`

[Optional] When specified, returns cell libraries for a specific library type. Otherwise, only nominal libraries are returned.

`-pvt <pvt_name>` Specifies the PVT name from which the cells libraries should returned.

-suffix <*string*> [Optional] When specified, returns a list of cell libraries for a specific PVT and library type with the specified suffix. Otherwise, cell libraries for a specific PVT and library type without suffix will be returned.

Note: This command is supported in the arc packet flow only.

Example

Consider that an LDBX post-process script writes out another output library with a margin added to delay tables and these output libraries have suffix "`_post_process`". The cell library files under output `ldb.gz/pvt1/LIBS/ directory` will be as follows:

```
cell1_pvt1.lib
cell1_pvt1.lib_ccs_tn
cell1_pvt1.lib_ecsm_tn
cell1_pvt1.lib_post_process
cell1_pvt1.lib_ccs_tn_post_process
cell1_pvt1.lib_ecsm_tn_post_process
```

```
cell2_pvt1.lib
cell2_pvt1.lib_ccs_tn
cell2_pvt1.lib_ecsm_tn
cell2_pvt1.lib_post_process
cell2_pvt1.lib_ccs_tn_post_process
cell2_pvt1.lib_ecsm_tn_post_process
…

In char.tcl:
write_library ….. output.lib
write_library ….. output.lib_ccs_tn
write_library ….. output.lib_ecsm_tn

if {[distr_on_client_assembly]} {
     set cell_libs [distr_get_onecell_libs]
    # running LDBX script on all libs of a cell and a pvt written out from a client.
     # in this case they are lib, lib_ccs_tn and lib_ecsm_lib three libs from this
cell and pvt
     ldbx_python script.py $cell_libs
     # other commands running with $cell_libs as input
}
if {[distr_on_master]} {
     foreach pvt [list pvt1] {
     set libtypes [list "lib" "lib_ccs_tn" "ecsm_tn"]
     foreach type $libtypes {
        # Assume directory contains files as above, i:
        # in each iteration command " distr_get_onetype_libs " would return list
of cell lib paths like following:
        # [cell1_pvt1.lib_post_process cell2_pvt1.lib_post_process …]
       # [cell1_pvt1.lib_ccs_tn_post_process cell2_pvt1.lib_ccs_tn_post_process…]
        # [cell1_pvt1.lib_ccs_ecsm_post_process
cell2_pvt1.lib_ecsm_tn_post_process…]
         set lib_list [distr_get_onetype_libs -pvt $pvt -libtype $type -suffix
"_post_process"]
         append_library -filename full_library_${pvt}.${type}_post_processed
$lib_list ${pvt}_post_processed
     }
}
}
```

em_buffer_cell_pin_bounds

Specifies a list of cells to buffer (see em_window_estimate_mode) that are known to have failed window estimation before.

Options

`-cells`	Specifies a list of the cells to buffer. Wildcards are supported. Default: * (all cells)
`-pins`	Specifies a list of the pins in the cells to buffer. Wildcards are accepted. Default: * (all pins)
`-indices (-1.0 -1.0)`	Specifies a list of the pairs of slew (nS) and load (pF) to buffer.

This command must be set before the `char_library` command is run.

Example

```
em_buffer_cell_pin_bounds  \
     -cells {SDFQOPTKS5D4BWP360H10P66PDP2ULVT} \
     -pins {Q} \
     -indices {0.0001 0.000002}
```

em_follow_hidden_power

Directs Liberate to characterize electromigration (EM) models for the specified pins. By default, Liberate does not characterize EM data for input pins.

Options

`-cells {`*`list`*`}`	List of cell names. Default: `*` (all cells)
`-pins {`*`list`*`}`	List of pin names. Default: `*` (all cell pins)

Use this command to specify input pins to characterize for EM using hidden power arcs. For this command to have any effect, EM characterization must be enabled (see the -em option of the char_library command and the Electromigration Models section of Chapter 7, "Performing Characterization using Liberate."

This command must be set before the `char library` command is run.

esource

Loads the specified file that was encrypted using the Liberate encryption utility (`alencrypt`).

Options

<encrypted_filename> Name of an encrypted file.

This command must be set before the `char_library` command is run.

find_critical_voltage_corners

Facilitates setting up of Critical Voltage Corners Selection (CVCS) flow that is designed to reduce number of voltage corners for characterization, but still maintain the accuracy. CVCS defines which voltages are the most representative from the given minimum-maximum voltage range (retrieved from the input libraries). Less number of corners to be characterized results in saving the time, resources, and storage, while smart way of corner selection ensures the accuracy.

Options

`-work_dir "string"` Specifies a directory to use as a workspace.
Default: `./cvcs`

`-cells {list}` Specifies a list of cells to analyze for critical corners (CC) search.
Default: use all cells

`-index_1 {list}` Specifies the `index_1` values to use for CC search. The position begins counting with `1`.
Default: use all indexes.

`-index_2 {list}` Specifies the `index_2` values to use for CC search. The position begins counting with `1`.
Default: use all indexes.

`-cc_db_file "string"`

Specifies the name of the database file to contain preprocessed data. If the file does not exist, a new file is generated for use in the next runs. Otherwise, it reads the file that is preventing data preprocessing. By default, no file is generated.

`-cc_file "string"` Specifies the name of the text file to contain a list of critical voltage corners. Default: `"cc.list"`

`-cc_method <herons | dynamic | linear>`

Specifies the method to find the inflection points.
Default: `herons`

`-cc_number <integer>`

Specifies the exact number of critical corners.
Default: number of CC is chosen automatically

`-cc_reltol <float>`

Specifies relative tolerance to use in inflection points search.
Default: `1.0`

`-cvcs_method "string"`

Specifies the Critical Voltage Corners Selection (CVCS) method to be used. The supported values include the following: `triangulation`, `sim_anneal`, `bisection`, `fibonacci`, and `uniform`.
Default: `triangulation`

`-interpolation_type "string"`

Specifies the interpolation method to be used. The supported values include the following: `linear` and `cubic`.
Default: `cubic`

`-abstol <float>`

Specifies the absolute tolerance to use in bisection-based CVCS.
Default: `0.001`

`-reltol <float>`

Specifies the relative tolerance to use in bisection-based CVCS.
Default: `0.01` (1%)

`-table_points "string"`

Specifies table points to be used for analysis. The supported values include the following: `3x3` and `all`.
Default: `3x3`

`-no_interpolation` Specifies not to interpolate and ignore the entries having different indexes. Only the matching ones are used.

`-plot_file "string"`

Specifies the name of the PDF file to contain clustering plots. By default, no file is generated.

`-pvt_file "string"` Specifies the name of the Tcl file that contains the `define_pvt` commands for critical PVT corners. By default, no file is generated.

`-sorted_weight_file "string"`

Specifies the name of text file to contain list of cells/arcs/points sorted by max weight. By default, no file is generated.

`{libFiles}` (Required) Specifies a list of input library files.

Critical Voltage Corners Selection

The Critical Voltage Corners Selection (CVCS) method supports the following algorithms:

- Fixed number of CVC; minimize interpolation error:
  - triangulation (that is, current CVCS)

- simulated annealing
- bisection
- uniform points (no input data processing)
- Fibonacci points (no input data processing)

- Unlimited number of CVC; minimize error up to some threshold:
  - triangulation
  - bisection

The CVCS method supports selection of table points to be used for analysis from `'3x3'` and `'all'` (controlled by `-table_points`).

Once `-cc_number` is set, CVCS looks for the best N corners to cover based on the specified voltage range with the best possible accuracy.

If `-cc_number` is not provided, then unlimited CVCS is started for finding minimal set of CVC and thus, providing good accuracy. Only the `triangulation` and `bisection` methods support unlimited CVCS. The default values for both these methods are:

- `triangulation`: Default relative tolerance is `1%` (controlled by `-cc_reltol`)
- `bisection`: default `abstol=0.001`, `reltol=0.01` (1%) are controlled by `-abstol` and `-reltol`, respectively

Example

```
# voltage scaling specific setting (needed to write out PVT file for critical
# corners)
set pvt_commands "set PROCESS tt\n set TEMP 70\n set VSS 0\n"
define_pvt_for_scaling -VLow 0.8  -VHigh 1.2 -Vincr 0.1 -VDD {VOLT VDD} \
    -pvt_commands $pvt_commands

find_critical_voltage_corners \
    -pvt_file "cc_pvts.tcl" \
    -cc_number 5 \
    -plot_file "cc_clustering.pdf" \
    -sorted_weight_file "cc_weights.txt" \
    $libs_list
```

generate_io_template

Generates a `template.tcl` file from a Microsoft Excel file that contains cell information and pin functions. This `template.tcl` file contains the define_cell and define_arc commands that are needed for the char_library `-io` flow.

Note: This command can read in only a Microsoft Excel file of `.xlsx` format.

Syntax

```
generate_io_template <input_excel_file_name> <output_template_file_name>
```

Options

`<input_excel_file_name>`

Specifies the name of the input Microsoft Excel file.

`<output_template_file_name>`

Specifies the name of the output template file.

This command should be set before the `char_library` command is run.

Example

In `char.tcl`:

```
generate_io_template RT_io_Char_Request_Form_DDR_4_all.xlsx template.tcl
source template.tcl
```

get_cells

Returns the list of cells in a session.

Options

`-type <netlist | leafcell | template | char | model | all>`

Specifies the type of cells to be filtered and returned. Valid types include: `netlist`, `leafcell`, `template`, `char`, `model`, and `all`. Default: `all`

List of cells for which a netlist has been read through read_spice. Returns an empty list prior to read_spice. If multiple read_spice commands are run, returns a superset of all netlist subckts read. This can include subckts that are hierarchical and will not be characterized.

`netlist` — List of cells for which a netlist has been read through read_spice. Returns an empty list prior to `read_spice`. If multiple `read_spice` commands are run, returns a superset of all netlist subckts read. This can include subckts that are hierarchical and will not be characterized.

`leafcell` — List of cells marked as leaf cells. The list includes cells identified as leaf cells using the define_leafcell command and auto-generated leaf cells during netlist parsing. To obtain "auto-generated" leaf cells when using Liberate Variety, all `read_spice` commands must precede the call to the `get_cells` command. For all other programs, the `get_cells` command must follow `char_library` (Liberate) and `char_macro` (Liberate MX).

`template` — List of cells from all define_cell commands that have a template (`define_cell` with a template), but ignores the driver-only cells (no template).

	`char`	List of cells that were characterized. Returns an empty list before the `char_library` command. Includes all cells that were read using read_ldb and all cells that are characterized in the current session.
	`model`	List of cells that were written into a `.lib` file in the last write_library command.
	`all`	List of cells including all of above (default).

`-status <passed | failed | all>`

Specifies the status of the cells to be filtered and returned. Valid status include: `passed`, `failed`, and `all`. Default: `all`

`passed`	List of cells that were preprocessed successfully.
`failed`	List of cells that were not preprocessed successfully.
`all`	List of all cells. Do not filter based on passed/failed status. (Default)

<regexp> Specifies the pattern to be used when matching cell name. Pattern is any regexp pattern. Default: no pattern, match all cells.

Examples

```
#Example 1:
    set cell_all [get_cells]
    puts "$cell_all"
Output:
    DFFX1 INVX1 NOR2X1

#Example 2:
    set cell_netlist [get_cells -type netlist]
    puts "$cell_netlist"
Output:
    DFFX1 INVX1 NOR2X1

#Example 3:
    set cell_template [get_cells -type template]
    puts "$cell_template"
```

```
Output:
    DFFX1 NOR2X1
```

get_pvts

Returns a list of PVTs that have been defined using the `define_pvt` command.

Note: The `get_pvts` command should be used after all the `define_pvt` commands because it can return only the PVT names that have been previously defined.

For more information, see Chapter 9, "Using the Multi-PVT Characterization Flow of Liberate Trio."

See also define_pvt and set_pvt.

Options

None

Example

```
set pvts [get_pvts]
foreach pvt $pvts {
    set_pvt $pvt
    set file_path "${rundir}/LIBS/${libprefix}_${pvt}.lib"
    puts "writing LIBFILE = $nom_file"
    write_library -filename $file_path $pvt
}
```

get_var

Returns the current value of the specified Liberate parameter, that is, the default value or the value set for the parameter using set_var.

Note: A list of supported Liberate parameters can be generated using the printvars command.

Options

<parameter_name>	Specifies the Liberate parameter name for which the current value is needed.

Example

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

get_var_default

Returns the default value of the specified parameter. The parameters are defined using the set_var command.

Options

<parameter_name>	Specifies the parameter name for which the default value needs to be returned.

help

Displays detailed description of the specified command or parameter in Cadence Help.

Note: This command will work only after you set the path to the LIBERATE installation directory. For more information, see Invoking Liberate Help.

Options

`{<command_name> | <parameter_name>}`

Specifies the name of the command or parameter for which help content needs to be viewed.

`-searchdoc <search_string>`

Displays search results for the specified strings in Cadence Help. Multiple search strings can be provided within double quotes.

Example

```
help -searchdoc "comparison library"
```

This command will display search results related to comparison and library.

interpolate_library

Generates an interpolate library at a new voltage corner using the given set of input libraries that have matching temperature and process values.

Note: By default, the `interpolate_library` command interpolates both table values and indexes. To use templates for applying custom indexes to the interpolated library, provide the required template using commands such as define_template and define_cell before running the `interpolate_library` command.

Options

`-voltage <float_value>`

Specifies the target voltage. It accepts a float value.

`-temperature <float_value>`

Specifies the temperature. It accepts a float value.

`-process <float_value>`

Specifies the process. It accepts a float value.

`-n_nearest <int_value>`

Specifies the number of nearest libraries to use for interpolation. It accepts an integer value. Default: `4`

Setting this option enables the following configuration:

`-n_nearest=2` --> linear interpolation

`-n_nearest=3` --> linear interpolation

`-n_nearest=4` --> cubic interpolation

-cells {<list>} Specifies a selective list of cells. Default: `{*}` (all cells)

-exclude {true | false}

Specifies whether to exclude or include the list of cells given with the `-cells` option. Default: `false`

Important Points to Remember

- Interpolation requires at least two input libraries to process.
- Target voltage must be intermediate value for input libraries voltages.

- Table indexes are also interpolated by default. You can apply custom indexes to resulting lib by setting templates (use define_template, define_cell, and define_index commands).

Example

```
read_library {in1.lib in2.lib in3.lib in4.lib}
interpolate_library -voltage 0.9 -temperature 70 -process 1 -n_nearest 2
write_library -filename out.lib interp_lib
```

merge_library

Takes two (or more) libraries and merges them to make a new library.

Important

Merging libraries should be performed as a separate Liberate run.

Options

`-arcs` Specifies that arc data should also be merged. By default, all library, cell, and pin data is merged.

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

Specifies a list of cells from the `from_lib` which should be merged into the `to_lib`. Default: all the cells in the `from_lib` should be merged.

Note: This option supports the use of a wildcard. If `-exclude` is used, the cells specified in the `-cells` list will be excluded and all the other cells included in the merge process.

`-driver_waveform` Preserves the driver waveform associations present in the “from” library.

Note: The `merge_library` command does not preserve any driver waveforms in the "to" library. By default, all references to driver waveforms are dropped from the output library.

`-exclude` Controls whether to exclude the cells specified with the `-cells` option from the second library.

Note: This option supports the use of a wildcard.

`-filename <`*`filename`*`>`

Output file name of merged library. Default: <to_lib>.m

`-indent <`*`number`*`>` Specifies the number of characters (spaces) to use for indentation in the merged library.

`-libname <`*`library_name`*`>`

Specifies the name of the merged (resultant) library.

`-method {`*`method data_type`*`}`

Specifies the method for merging data into a library. Default: `"append"` (Append data missing from the target library.)

By default, `merge_library` appends only data that is missing in the first library from the second library. With the `-method` option, you can select the `min`, `max`, `avg` (average), `append`, and `sum` of data for any arcs that are present in both libraries to create the merged library. For example, `"max"` or `"delay avg hold max"`.

The options are specified as a list of method and type pairs. Legal data types for this method of merging are `delay`, `retain`, `power`, `setup`, `hold`, `recovery`, and `removal`.

The following example shows how to merge two libraries and use the max data for the hold time:

```
merge_library -method "max hold" -filename mlib lib1 lib2
```

`-no_extra_pins` Disables merging into the `to_lib` any pins that do not already exist in the `to_lib`. This means that during the merge operation, the extra pins are excluded from the second library.

`-type {`*`list`*`}` Merges the specified types. Supported types are: `attributes`, `cap`, `ccs`, `ccs_cap`, `ccs_retain`, `ccsn`, `ccsp`, `constraint`, `delay`, `ecsm`, `ecsm_cap`, `ecsmn`, `ecsmp`, `em`, `hold`, `leakage`, `max_cap`, `max_trans`, `nochange`, `noise`, `min_pulse_width`, `power`, `recovery`, `removal`, `retain`, `setup`, and `minimum_period`. Default: Merge all types.

`-unique_pin_data` Outputs unique data, such as timing and power, for each bus bit or bundle member (similar to `-unique_pin_data` for write_library). It specifies that original pin names are to be used inside the *when* condition string without going through the post-processing of changing pin names to bundle names.

Without unique_pin_data	*With* unique_pin_data

bundle (QB) {
  pin (Q0B) {
  ...
  }
  pin (Q1B) {
  ...
  }
  timing () {
  ...
  }
}
	bundle (QB) {
	  pin (Q0B) {
	  ...
	  timing () {
	  ...
	  }
	  }
	  pin (Q1B) {
	  ...
	  timing () {
	  ...
	  }
	  }
	}

`-voltage <voltage>`

Instructs Liberate to create a new library that contains values calculated by performing voltage scaling from two existing libraries. All characterized timing data in the library is scaled including `delay`, `transition`, `constraints`, `retain`, `ccs`, `ccs_retain`, and `ecsm`. In addition, internal power, leakage, and capacitance (pin cap and receiver capacitance (`ccs`, `ecsm`)) values will be scaled. Current-based power (`ccsp` or `ecsmp`) and noise data (`si`, `ccsn`, or `ecsmn`) will ***not*** be scaled and will be copied from the first library.

A comment is inserted in the output library to indicate that merging with voltage scaling was performed.

For example:

```
merge_library -voltage 1.0 -filename newLib_1.0.lib
libA_0.9.lib libB_1.1.lib
```

`<to_lib>`

Specifies the library that will form the initial data in the new library. This is the "target" library into which data from the `from_lib` will be merged.

{<*from_lib*>}

Specifies one or more libraries from which data will be taken and merged into the "target" (to_lib). If more than one library is specified, the merging takes place sequentially, in the order that libraries are specified.

The libraries in the from_lib list can contain wildcards in the cell name if surrounded by double quotes. This can be used, for example, to merge the footprint attribute "AND" to all of the cells whose names are prefixed with AND. The to_lib does not support the use of wildcards. An example of wildcard usage in a from_lib library is as follows:

```
cell ("AND*") { footprint : "AND" }
```

Note: If merging data from complete libraries (e.g. no changes are expected), it is recommended to set adjust_tristate_load to 0 prior to merging.

To merge multiple libraries, the merge_library command starts with the to_lib, and merges data from the from_lib into it.

Note: The to_lib will ***not*** be overwritten; the merge process takes place in memory and the results of the merge will be written to the file specified with the filename option.

All the data in the to_lib will be preserved and any extra data in the from_lib will be added. If a list of from_lib libraries is provided, the libraries will be merged into the output library, one library at a time. For example, any cells in the from_lib that are missing from the to_lib will be combined into the merged library and the merged output library will be sorted alphabetically. The merge_library command also supports bus structures in the lib file, meaning that Liberate will keep buses in the merged library if the source library has buses.

Normally, merge_library does not replace data; it only appends it. (See option method for an exception to this.) For example, if a to_lib has NLDM data and a from_lib has NLDM+CCS data, only the CCS data in the from_lib will be merged; the NLDM data in the to_lib will not be changed. Similarly, if a list of from_lib are given with overlapping data sets, only the first instance will be merged.

The merge_library command can also be used to add any missing attributes and groups, such as CCS, CCSN, and ECSM data, to the equivalent cells in the to_lib. For libraries that are not created by Liberate, merge_library will attempt to find an equivalent arc in the to_lib. An equivalent arc will have the same pins, data type and attributes (e.g. timing sense, timing type etc.) and an overlapping or equivalent logic state. If no match is found a warning is generated, indicating that the data could not be merged with the target library.

Example

```
# Merge CCS into an existing library
merge_library -filename orig_ccs.lib orig.lib ccs.lib
```

one_cold

Accepts a list of pins and returns a string that can be used to specify the one-cold relationship of the pins. For example, `one_cold { a b c }` will be converted to `((!a b c)+(a !b c)+(a b !c))`.

Note: This is a Tcl function.

Options

`{ list }`	Specify a list of one-cold pins (Default: none)

This command must be used before char_library.

Example

```
define_cell -when [ one_cold { a b c } ] …
```

one_hot

Accepts a list of pins and returns a string that can be used to specify the one-hot relationship of the pins. For example, `one_hot { a b c }` will be converted to `((a !b !c)+(!a b !c)+(!a !b c))`.

Note: This is a Tcl function.

Options

`{ list }` Specify a list of one-hot pins. Default: `none`

This command must be used before char_library.

Example

```
define_cell -when [ one_hot { a b c } ] …
```

packet_slave_cells

Returns a unique list of cells assigned to the client, also known as, slave.

Note: This is a Tcl function.

When called from the server, an empty list is returned. See Parallel Processing.

Options

None.

Example

```
# Set the constraint_delay_degrade global parameter to 8% for myCell in the client
only
if { [packet_slave_cells] == "myCell" } {
    set_var constraint_delay_degrade 0.08
}
```

parallelize_tasks

Submits all the tasks created using the create_task command and runs the tasks in parallel on the clients. This command should be used along with the `create_task` command in the packet arc flow with Bolt job distribution system.

Note: To parallelize different tasks that are dependent on each other, you can use the `create_task` and `parallelize_tasks` command sets multiple times in the same session of Liberate. This helps to leverage the Bolt job distribution system and reduce the TAT by parallelizing the task at per PVT or per cell level and utilizing multiple CPUs instead of just one.

Options

`-workdir <log_directory_name>`

(Optional) Specifies a working directory for running all the parallelized tasks and saving the client log files.

When specified, the directory hierarchy is created; otherwise the client log files are saved under the current run directory.

Example

```
##### Example of run.tcl script that creates and parallelizes tasks to run
compare_library on cell level #####
set rootdir [pwd]


# setup bolt
set_var packet_arc_job_manager bolt
set_var packet_clients 5
set_var rsh_cmd "bsub -q liberate -R \"(OSREL==EE50||OSREL==EE60) rusage\[mem=1\]
span\[hosts=1\]\" -o %B/%L -e %B/%L"
set_var packet_client_health_checks 1


source ${rootdir}/cells.tcl


# create tasks to run in parallel
set pvts {noms}
foreach pvt $pvts {
    foreach cell $cells {
       create_task -script $rootdir/compare_library.tcl -args [list $rootdir $pvt
$cell] -logdir $pvt/$cell
```

```
    }
}
# submit all tasks just created from create_task and runs them in parallel on
clients
parallelize_tasks -workdir "./clientlogs"

###### example of task script compare_library.tcl #########
#####----------------------------------------
##### receive arguments
#####----------------------------------------
set rundir [lindex $argv 0]
set pvt [lindex $argv 1]
set cell [lindex $argv 2]

#####----------------------------------------
##### set reference and compare libraries
#####----------------------------------------
set ref ${rundir}/DATA/lib/example_nldm.lib
set cmp ${rundir}/DATA/lib/example_nldm_new.lib

#####----------------------------------------
##### parallel compare_library
#####----------------------------------------
compare_library -cells $cell -report ${rundir}/$pvt/$cell/${cell}_cmp.rep $ref
$cmp
```

printvars

Lists the current values of all of Liberate command parameters.

Options

None.

Example

```
# List Liberate parameters
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 = /.
toggle_leakage_state = 0
```

read_ldb

Reads an existing library database (ldb) created by the write_library command. The library database can then be used for formatting the library data, for example, creating a datasheet.

This command supports specification of additional context bins in the following situations:

- *Situation 1:* `N` bins were characterized originally, and now `N+2` bins are needed.
- *Situation 2:* A fresh library was characterized originally, and now `N` bins are needed.
- *Situation 3:* `N` bins were characterized originally, but bin `0` to `N-1` passed while bin `N` has failures. Now, bin `N` needs to be recharacterized.

Options

`-check_driver_waveforms`

Checks both pre-driver and active-driver generated waveforms in ldb to verify that they have the correct slews.

`-check_spice`

Checks netlist and model for changes and recharacterize cell if any changes are detected.

`-check_var`

Checks parameters and commands for changes. If a change is detected, that change will map to a specific `remove_type`, and that type (or types) will be recharacterized. This works only if `-incremental` is set.

`-incremental`

Specifies incremental capability. This option instructs Liberate to examine the ldb and performs characterization for any data that is missing from the LDB.

Note: The `-incremental` flow is currently supported when running a single cell and is not supported with any packet flow (see packet_clients).

`-pvt` Specifies the single PVT corner to read from the characterization database (LDB directory). The LDB must have been created using the multi-PVT characterization flow.

This option accesses a specific corner LDB directory within a multi-PVT LDB network location.

For more information, see Chapter 9, "Using the Multi-PVT Characterization Flow of Liberate Trio."

See also Example 2 and Example 3 to know how to use the `-pvt` option with the `read_ldb` command in multi-PVT characterization flow.

`-remove {list_of_cells}`

Specifies a list of cells to remove from the ldb. Default: (none)

By default, Liberate will not characterize any cell that has already been loaded from an ldb. A cell that is removed from the ldb during loading will be recharacterized.

`-remove_failed` Specifies to automatically remove cells that are marked as failed in the ldb. In most flows, this can be used in place of "`read_ldb -remove { list of failing cells }`".

`-remove_type {list_of_types}`

Specifies a list of data types to be removed from the ldb, so they will be recharacterized. Supported types are: `ccs`, `ccsn`, `ccsp`, `cin`, `constraint`, `delay`, `hold`, `leakage`, `mpw`, `power`, and `setup`. Default: `none`

`<filename>` A library database file in ldb format.

Liberate can also use an existing ldb to recover from any characterization run that didn't complete successfully. As each cell is characterized it is saved to an ldb in the current directory, named *altos.ldb.<#>.* This temporary ldb can subsequently be read by Liberate (using `read_ldb`) to complete the characterization. A complete ldb that contains all the cells can then be saved using `write_ldb`.

Important

You may have a given parameter stored in the ldb and also specified in your Tcl script. Whichever occurs *last* in your script will determine the value of the parameter (that is, the last setting of the parameter will override an earlier setting.)

The `read_ldb` command will automatically recognize if the input file is gzipped during loading. The GNU gzip utility must be in the search path if the input file is gzipped.

Using the -check_spice option

`check_spice` enables checking if the spice netlist and/or the spice model has changed. The Spice netlist check is based on the parsed X/M/D/R/C circuit elements, which generates a checksum, along with a count and total value of each element type that is stored in the ldb. The Spice model check is based on the top-level model path name & time stamp, which is stored in the ldb. If a change in the netlist or models is detected, the cell will be removed and completely recharacterized. Required for this flow are:

- set option `-incremental`
- set parameter enable_command_history
- set parameter rechar_chksum (specifies the linux routine to generate a checksum.)

Suggested uses for incremental capability:

- Adding or deleting arcs; only the affected arcs will be re-simulated.
- Arcs containing errors from a previous characterization (altos_error_flag in the ldb) will be automatically recharacterized.
- Re-simulating only constraint arcs (for example, if constraint_delay_degrade was changed). This will be faster than recharacterizing the entire library:

```
read_ldb -remove_type {constraint}
char_library
```

- Adding CCSN to a previous ldb that does not contain CCSN data. Use these commands:

```
read_ldb -incremental
char_library -ccsn
```

  CCS timing and/or power can be handled similarly.

- Add additional context bins. Consequently, the following situations might arise:

  Situation 1:

  N bins were originally characterized, now *N*+2 bins are needed.

  Situation 2:

  A fresh library was originally characterized, now *N* bins are needed.

  Situation 3:

N bins were originally characterized, but `Bin 0` to *N*`-1` passed while bin *N* has failures, now bin "N" will be re-characterized.

Examples

Example 1

```
# Read an ldb to generate additional formats
read_ldb tt_all.ldb
# Write .lib
write_library tt_all.lib
# Write a HTML datasheet
write_datasheet -format html -dir tt_html tt
```

Example 2

```
# Use of -pvt in Multi-PVT Characterization Flow
# Load one corner from an existing LDB with multiple PVT corners
read_ldb -pvt cbest_max_0c $path_to_mpvt_ldb
```

Example 3

```
# Use of -pvt in Multi-PVT Characterization Flow
# Load one corner and remove a list of cells prior to recharacterization
read_ldb -pvt cbest_max_0c -remove $list_of_cells $path_to_mpvt_ldb
```

read_library

Reads an existing Liberty format library into memory.

Options

`-cells { <cell_names> }`

(Optional) A list of cell names. This list can include wildcards.

When specified, the library being read is first broken down (in the memory) into a smaller library that contains only cells specified for improved read performance. This feature is supported only for libraries produced by Liberate.

You can use this feature along with the parallelize_tasks command where the command in parallel requires a cell-level library to be read in, when there is only a large library with many cells available. This can significantly speed-up the parallel `read_library` jobs.

`{library_names}` List of library file(s) to read in Liberty format.

This is a key command for a number of flows capable of outputting these types of files:

- **User data file** - Extracts non-characterized data such as area, function, etc.
  See write_userdata_library
- **Template file** - Tcl script used as part of a characterization flow
  See write_template
- **Datasheet** - Typical datasheet showing delays, power, pin capacitance, etc.
  See write_datasheet
- **Verilog / Vital** - RTL model descriptions
  See write_verilog and write_vital

Examples

Example 1

```
# Create a datasheet
read_library cdnTech1.lib
write_datasheet -format text myDatasheet
```

Example 2

`read_library` can also be used together with `write_library` to add margin or other attributes to a library:

```
# Add 20% relative margin to setup and hold
read_library cdnTech2.lib
add_margin -type {setup hold} -rel 0.20
write_library test.lib
```

Example 3

When a user wants to read a library with bus syntax “<>” and would like to use the output to generate another bus syntax (that is, “[]”). The below commands in the same sequence will work. In this example the bus syntax is “<>” and the library is written with “[]”.

```
set_var bus_syntax "<>"
read_library input.lib
write_library -bus_syntax "\[\]" -filename output.lib test1
```

read_spice

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

Important

It is recommended that you do not mix the netlist formats. For example, do not mix a SPICE netlist with a Spectre model file or a Spectre netlist with a SPICE 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.

The `read_spice` command supports circuit components such as resistors, capacitors, and metal–oxide–semiconductor field-effect transistors (MOSFETs). A zero volt DC voltage source will be replaced by a `0.001` Ohm resistor during the internal flattening of the circuit to facilitate the logic analysis by the *Inside View* algorithm. If `extsim_flatten_netlist = 1` (default), the simulation will include this `0.001` Ohm resistor in place of the voltage source. If `extsim_flatten_netlist =0`, the simulation deck will include the original netlist. All other element types are not supported by the *Inside View* algorithm.

Options

`-dspf_floating_caps`

Improves the load time of a netlist if it is in the DSPF format and it exceeds 500 MB.

Note: This option is compatible only with Liberate MX and Liberate AMS dynamic mode.

`-format { spice | spectre }`

Specifies the SPICE netlist format.

`-parser { native | sfe }`

Specifies the parser to use when reading in netlist and model files. Default: `""` (Instructs Liberate to use the SFE parser for Spectre netlist types.)

See the parse_sfe_parser_mode parameter to change the default behavior.

`{ <spice_netlist_file> }`

(Required positional option) List of file(s) with extracted circuit netlists in SPICE format, including models.

The model files should be included with the circuit netlists. The SPICE netlist and model formats supported by Liberate are as follows:

- Spice3 netlist, BSIM3 and BSIM4 models. Well proximity effects are supported.
- PSP

This command must be used before the char_library command is run.

Examples

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

read_training_data

Reads the specified training database files or regular CCSN libraries for fixing bad DC tables in a characterized library.

Note: A training database is an encrypted file that contains normalized DC tables.

Options

`{<filename>}`	Input a list of training database files or regular CCSN libraries. **Note:** If regular CCSN libraries are specified, a training database is generated based on them.

The `read_training_data` command must be used before running the write_library command.

You can use your own pre-generated training databases with the `read_training_data` command. If you do not want to update the existing training database and want to write out another one instead, then use the write_training_data command in Liberate. This command generates a new training database based on the library that the write_library command outputs.

To ensure that your training database is updated with good tables found during the CCSN DC table checks, set the update_training_data parameter to `1` before running the `write_library` command.

Examples

1. DC table fixes:

```
read_training_data $dbList
read_library $lib
write_library … -ccsn -fix_dc $lib
```

2. Generate a new training database / merge existing databases:

```
# Enable to read both training databases and pure .lib files.
read_training_data $dbList ;
# Liberate reads and checks the data, and then creates one new database.
write_training_data $newDb
```

3. Update the training database on the fly while checking the `write_library` data:

```
read_training_data $dbList
set_var update_training_data true
```

```
read_library $lib
write_library … -ccsn -fix_dc $lib
write_training_data $newDb
```

read_truth_table

Reads in and validates truth-table files.

Options

{<*filename*>} Specifies a list of Truth Table files that contain one or more truth tables to load.

This command must be used before running the write_template command. Use the `write_template` command with the `-truth_table` and `-auto_index` options to output a template file built from the loaded truth table file data.

The define_pin_load template 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, they need to be added 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.

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
* CELL_END
```

Example output template:

```
Cell: PDIO12
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

define_leakage -when "!OEN !I" {PDIO12}
define_leakage -when "!OEN I" {PDIO12}
define_leakage -when "OEN I !PAD" {PDIO12}
define_leakage -when "OEN !I !PAD" {PDIO12}
define_leakage -when "OEN I PAD" {PDIO12}
define_leakage -when "OEN !I PAD" {PDIO12}
define_leakage -when "OEN I" {PDIO12}
define_leakage -when "OEN !I" {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 -> C
# 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 \
}
```

read_vdb

Reads in a VDB file that was previously created by the write_vdb command. The Tcl file must contain the same setup that was in place when the VDB file was created.

Options

`{<filename>}` The name of a VDB file to load.

Use the `read_vdb` command in a Tcl command file before the char_library command. When using a `read_vdb` command, do ***not*** reference a `template.tcl` file.

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

reset_defaults

Restores the changed parameter settings to the default values from a previous release.

Options

`-version <version>` Specifies the release from which the parameter settings need to be restored.
Default: Reset all parameters needed to restore the behavior in the previous release.

From release to release, defaults values of some Liberate parameters might change. You can use this command to restore the default values from a previous release. The required default value changes are stored in the file located at the following location:

```
${ALTOSHOME}/etc/backward_compatible.tcl
```

The backward compatible settings echoes into the Tcl file.

Important

This command should be used at the start of your Tcl file and can only be used once per run.

Example

```
# Reset Liberate defaults to 12.1
reset_defaults -version 12.1
```

select_arc

Specifies the arc to be used for simulation. This command is useful for isolating arc(s) from the previous characterization run during a debugging session.

Note: Only the arcs defined using define_arc can be specified with `select_arc`.

Options

`-type { combinational | edge | async | enable | disable | retain | power | hidden | setup | hold | recovery | removal | non_seq_setup | non_seq_hold | nochange_low_low | nochange_low_high | nochange_high_low | nochange_high_high | mpw | minperiod | min_period | min_clock_tree_path | max_clock_tree_path | ccsn_first | ccsn_last }`

Specifies the type of arc. Default: `combinational`

`-when <function>`

Specifies the logic conditions of the other pins of the cell to enable the specified arc using the Liberty `when` syntax.

Note: This option corresponds to the Liberty `when` attribute.

`-pin {<pin_list>}` Specifies a list of destination pins for the arc.

`-pin_dir <R | F>` Specifies transition direction of pins. This option can have one of the following values:

`R`	Specifies a rising transition.
`F`	Specifies a falling transition

`-related_pin {<pin_list>}`

Specifies a list of related pin names.

`-related_pin_dir <R | F>`

Specifies transition direction of the related pins. This option can have one of the following values:

`R`	Specifies a rising transition.
`F`	Specifies a falling transition

`-probes {<node_list>}`

Specifies a list of monitor node names.

`-points {<list>}` Specifies a list of points that need to be characterized. For example, `{{3 3} {4 4}}` or `{{1} {2}}` for MPW.

`{<cellNames>}` (Required) Specifies a list of cell names.

`select_arc` supports fuzzy search. In the following example, `select_arc` selects all the arcs that meets the specified conditions including, `pin_dir=R/F` and `related_pin_dir=R/F`, which are defined with `define_arc`.

```
select_arc -pin A -related_pin B -type combinational CELLNAME
```

This command must be specified before the char_library command is run.

select_index

Reduces the indexes to characterize without modifying the template file (see define_template and define_index).

Use this command in a debug flow to select specific data points to be characterized and reported.

Note: For variation characterization in Liberate Variety or in unified characterization flow of Liberate Trio, the minimum dimension of a template is 2x2.

Options

`-align_fastsim <boolflag>`

Aligns `index_1` and `index_2` with `slew_load` specified for FastSim run to perform simulation. Default: `false`

Note: This option is supported only in Liberate MX.

`-index_1 { list }`

Lists the specific `index_1` positions that should be characterized. The actual values are specified in the template file by using the `define_template` and `define_index` commands. The counting of the position begins with `1`. Default: all indexes (that is, indexes are not reduced).

The list of positions should be specified within curly braces, such as `{ 2 3 5 }`.

`-index_2 { list }`

Lists the specific `index_2` positions that should be characterized. The actual values are specified in the template file by using the `define_template` and `define_index` commands. The counting of the position begins with `1`. Default: all indexes (that is, indexes are not reduced).

The list of positions should be specified within curly braces, such as `{ 2 3 5 }`.

`-row_and_column <boolflag>`

Selects the entire row and column for simulation and interpolates other points in the lookup table. Default: `false`, selects the crossing point.

Note: This option is supported only in Liberate MX.

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

`"1x1"`	Uses the first index value. This is equivalent to `"-index_1 {1} -index_2 {1}"`. This setting is not supported in Liberate Variety and is overridden to `2x2`.
`"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}"`.
	`"index"`	Uses the settings of the `-index_1` and `-index_2` options.
	`"mid"`	Uses the middle index value. For an index with seven values, this is equivalent to: `"-index_1 {4} -index_2 {4}"`.
`-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_library command is run.

If there is more than one global `select_index` command (without the `-type` option), only the first one that is executed will be applied (not the last one). If there is more then one type of specific `select_index` command (with the `-type` option), only the first one that is executed will be applied. A type-specific `select_index` command will always override a global `select_index` command for the specified type.

When using `select_index` for debugging constraints, review the setting of the `constraint_output_load` parameter. If it is set with a syntax of `index_<num>`, it might not map to the desired load. A workaround is to set it to a specific load value.

set_aging_criteria

Enables Liberate to use the Spectre circuit aging capabilities. Before using this command, ensure that you are using MMSIM12.1 ISR16 or a later release. Check the `${ALTOSHOME}/README` file for the recommended version of MMSIM. You might get incorrect aging characterization data if you use an earlier version of the MMSIM release. In addition, use the extsim_model_include parameter and the define_leafcell command with `set_aging_criteria`.

Note: The aging models are required for the aging flow to work.

Options

`-aging_supply list{}`

Specifies the voltage(s) at which aging is run. It accepts a single value (in volts) applied to all supplies or a list of `supply_name` voltage pairs. The default is the voltage specified by the set_operating_condition command.

`-aging_temp <temperature>`

Specifies the aging temperature in degree Celsius. The default is the temperature specified by the set_operating_condition command.

Note: The `-aging_supply` and `-aging_temp` options allow you to run the aging analysis at a different voltage and temperature than used in the characterization.

`-attribute <value>` Settings about reliability analyses. (REQUIRED)

The `-attribute` option specifies the aging-specific settings for the Spectre reliability analysis. The following attributes should be included: `age time`, `deltad value`, and `report_model_param value`.

For more information on the aging-specific attributes, refer to the *Virtuoso Spectre Circuit Simulator and Accelerated Parallel Simulator User Guide*. There are no default attributes. This option is required.

`-duty_cycle <value>`

Specifies the duty cycle that is applied to the period for stress simulation. The stress simulation duty cycle ranges from `0.1` (10% of period at a high logic level) to `0.9` (90% of the period at a high logic level). Default: `0.5` (50%)

If this option is not specified, the default value of 0.50 (50%) is applied to netlist aging.

`-flatten_netlist` Specifies whether the stress file is generated using a flattened netlist.

Default: `false`

`-period <value>`

(REQUIRED) Specifies the period of stress simulation (in MKS units). Default: `0`

Note: This option is applicable only when `-type` is set to `both`.

If you set a period value that is less than `max_slew` or `duty_cycle`, Liberate corrects it as necessary. An information message is also displayed.

```
"INFO (LIB-1011): (set_aging_criteria): ... ..."
```

`-stress_file <string>`

Specifies the name of the stress file. Default: `none`

Note: If `-type` is set to `aged`, the `-stress_file` option must also be set.

`-stress_file_dir <string>`

Specifies the name of the directory containing the stress files.

Default: `stress_files`, if `-type` is set to `stress` or `cell_both`.

Note: This option is required if `-type` is set to `aged`.

`-type <string>`

Specifies the reliability type.

Supported values are: `both` (arc-level), `aged` (cell-level), `stress` (cell-level), and `cell_both`
Default: `"both"`

- `both`: Enables arc-level aging flow that first stresses the special arcs and then characterizes the arcs.
- `aged`: Enables cell-level aging flow and loads a user-defined stress file. This flow supports user-defined cell input stimulus and performs stress simulation. Liberate loads the existing stress files and then characterizes the arcs.
- `stress`: Enables cell-level stress simulation. Liberate generates the stress stimulus as defined in set_stress_criteria, and performs stress simulation.
- `cell_both`: Enables cell-level automatic aging flow that produces results equivalent to setting the `-type` option to `stress` and `aged`.

Note: If `-type` is set to `aged`, the `-stress_file` option must also be set.

This command must be used before the char_library command is run.

Examples

Example 1

```
set_aging_criteria \
    -period 200e-9 \
    -duty_cycle 0.5 \
    -aging_supply 1.375 \
    -aging_temp 125 \
    -attribute "age time = \[10y\]\n deltad value = 0.1\n report_model_param
    value=yes\n tmi_aging_mode type=aging\n agelevel_only value=\[1 2\]\n
    alter_flag alter param = alter_flag value=1"


rel-block in sim.sp:
rel reliability {
    age time = [10y]
```

```
    deltad value = 0.1
    report_model_param value=yes
    tmi_aging_mode type=aging
        tran_fresh tran  start=0  step=1e-12  stop=2e-07
        tran_aged tran  start=0  step=1e-12  stop=1e-07
}
```

Example 2

```
set_aging_criteria \
    -period 200e-9 \
    -duty_cycle 0.5 \
    -attribute "age time = \[10y\]\n deltad value = 0.1\n report_model_param \
        value=yes\n" \
     -type aging \
     -stress_file "[pwd]/DATA/stress" \

rel-block in sim.sp:
rel reliability {
    age time = [10y]
    deltad value = 0.1
    report_model_param value=yes

    simmode type=aging tmifile="xxx/inv.tmiage0"
    tran_aged tran  start=0  step=1e-12  stop=1e-07
}
```

set_attribute

Adds, replaces, deletes, or modifies the specified attribute in the output library.

Note: This command supports use of bus syntax. However, use of wildcard characters is not supported.

Options

`-cells { <cell_names> }`

A list of cell names. This list can include wildcards. Note: When you specify `-cells`, the attribute is set at the specified cell level. However, if you do not specify any of the location options (that is, `-cells`, `-pins`, `-related`, and `-group`), the attribute is set under the library level.

`-pins { <pin_names> }`

Specifies a list of pin names on which the attribute is set.

Note: If this argument is specified, the `-cells` argument is required.

`-type <type>` The type of attribute. Specify one of the following valid values: `constraint`, `delay`, `em`, `leakage`, `min_pulse_width`, `power`, or `retain`. Default: `""` (Do not apply this option.)

`-related <related_pin>`

The related_pin of an arc. This is used to identify a timing or internal_power group under the specified pins in the specified cells where the specified attribute will be set. Default: "" (Do not apply this option.) Note: If `-related` is specified, the `-cells` and `-pins` options are required.

`-group_name <group_name>`

The group name under which the attribute will be set. Examples: "ff" and "latch". Default: "" (Do not look for a matching group). If `-group_name` is not specified, the attribute is set under the level specified by the other options.

`-match_attributes {<attributes>}`

A list of existing attributes and their values that exist in the library. This is used to identify a specific location to set the attribute. For example, {timing_type rising_edge timing_sense positive_unate}. Default: "" (Do not match any existing attributes.)

`-method <method>` The method to be used for setting the attributes. Specify one of the following valid values: append, augment, replace, or delete. Default: replace (Replaces the existing attribute and adds it if it does not already exist.)

`-user_defined_type < string | integer | float | Boolean >`

Specifies the type of this attribute as specified in the `define` command in the library header. Default: `""` (no `user_defined_type`)

<attribute_name> The name of the attribute that needs to be edited.

<?attribute value?>

(Optional) The new value of the specified attribute. The value is an arbitrary string and should contain all required punctuation such as quotes. Note: If the -method option is to delete, do not provide a value.

The use of this command can create a library that does not compile or is functionally incorrect. This is because there is minimal checking of any attribute changes applied by using the `set_attribute` ***command. In addition, the set attribute should have the legal Liberty syntax, otherwise the final library will not compile.***

The `set_attribute` command allows you to add or modify attributes in the output library at the library, cell, pin, and group levels. In addition, you can add or modify attributes in groups of the following types: `delay`, `constraint`, `power`, `em` (electromigration), `leakage`, `min_pulse_width`, and `retain`.

A `test_cell` attribute added using the `set_attribute` command always takes precedence over the ones added using the `write_library -user_data` command and through automatic `test_cell` generation (see auto_test_cell).

How the attributes are applied depends on the value specified with the `-method` option. The `replace` method, which is the default, replaces an existing attribute value or adds it if the attribute does not exist. The `append` method appends additional copies of the attribute if it already exists; otherwise, it adds the attribute. The `augment` method adds an attribute only if the attribute does not already exist. The `delete` method removes the attribute and its value.

By default, the specified attribute is set at the library level. Use the `-cells` option to specify a list of cells (including wildcards) on which the attribute needs to be set. To set the attribute at a pin level, use both the `-cells` and `-pins` options.

Attributes that belong to a specific library, cell, or pin group can be set by specifying the `-group_name` option.

Attributes that appear under characterized arcs (such as `delay`, `leakage`, `constraint`, and `power`) can also be set using the appropriate `-type`, `-related`, and `-match_attributes` list.

For example, while editing attributes for a delay arc, the following options must be specified:

- `-cells <cell_name>`
- `-pins <pin_name>`
- `-related <related_pin>`
- `-type "delay"`

If a specific delay arc is to be edited, a list of attributes can be specified with the `-match_attributes` option. The specified attributes list uniquely identifies the arc to be edited, such as a specific `when` condition and `timing_type`.

Example

```
# Add the "nom_process" attribute with value 3 to the library level if it does not
# already exist
set_attribute -method augment nom_process 3

# Add the "default_hidden_power" attribute with value true to hidden power arc under
# pin B1 of cell AOI33 with the when condition "(A2 * !(A3) * B1 * B2 * !(B3) * ZN)"
set_attribute \
     -cells "AOI33" \
     -pins "B1" \
     -type power \
     -match_attributes {when "(A2 * !(A3) * B1 * B2 * !(B3) * ZN)"} \
     default_hidden_power \
     true

# Add the "setup_mode" attribute with value "max" to a constraint arc under pin D of
# cell SDFF to all entries with timing_type setup_rising.
set_attribute \
     -cells SDFF \
     -pins D \
     -type constraint \
     -match_attributes {timing_type setup_rising} \
     setup_mode \
     \"max\"
```

```
# Change the value of the min_pulse_width_low attribute to 0.040 under pin CP of
# cell CKGATE.
set_attribute \
     -cells CKGATE \
     -pins CP \
     -type min_pulse_width \
     min_pulse_width_low \
     0.040

# Change the timing_type of the constraint arcs under pin D of cell SDFF from
# hold_rising to hold_falling
# Note: This example shows the flexibility of the set_attribute command. It
# lets you make arbitrary changes to the attributes in a library. But when used
# incorrectly, as it is in this example, it can break a library.
set_attribute \
     -cells SDFF \
     -pins D \
     -type constraint \
     -match_attributes {timing_type hold_rising} \
     timing_type \
     hold_falling

# Change the function attribute of pin Y for all cells prefixed with AND2 to "A*B"
set_attribute -cells AND2* -pins Y function "\"A*B\""

# Remove the direction attribute from pin Y for all cells prefixed with OR2
# Note the resulting library will not compile.
set_attribute -method delete -cells OR2* -pins Y direction

# Add the test_cell group complex attribute
set_attribute -cells {SDFFX1} \
    test_cell { () {
        pin(SI) {
           direction : input;
           signal_type : "test_scan_in";
        }
        pin(D) {
           direction : input;
        }
        pin(SE) {
```

```
            direction : input;
            signal_type : "test_scan_enable";
         }
         pin(CP) {
            direction : input;
         }
         pin (Q) {
            direction   : output;
            function    : "IQ";
            signal_type : test_scan_out;
         }
         pin (QN) {
            direction: output;
            function: "IQN";
            signal_type : test_scan_out_inverted;
         }
         ff("IQ","IQN") {
            clocked_on : "CP";
            next_state : "(D&!SE) + (D&SI) + (SE&SI)";
         }
    }
}
# Set the attribute-value pair (abc :xyz) for pins Q[3], Q[2], Q[1], and Q[0]
set_attribute -cells BSPA76P302052ZZAZZ -pins "Q\[3:1\] Q\[0\]" abc xyz
```

set_client

Specifies a client a machine to be used for distributed library characterization.

Options

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

Defines a directory on the client machine to use as a temporary workspace for simulation jobs performed on that machine. Liberate creates the directory if it does not exist. (REQUIRED)

<machine_name> Name of client machine or queue name.

Liberate can 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 3, “Parallel Processing.”

This command must be used before char_library.

Examples

```
# set the machines to use (no queue)
set_client -dir /tmp/scratch/%U_%N_%S_%P linux1
set_client -dir /tmp/scratch/%U_%N_%S_%P linux2
```

The `-n` option supports a deprecated method of using a queuing system such as LSF. For more details on this option, see Appendix F, “Deprecated and Backward Compatibility Commands and Parameters.”

set_conditional

Disables conditional arcs for delay and constraint groups for a list of cells. Currently, other arc types such as power are not supported.

Options

`-off` Disable conditional states

`-type {delay const}`

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

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

List of cell names. (REQUIRED)

This command can be used after char_library, but before model generation.

Example

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

set_constraint

Adds the specified margin (in seconds) to a supported constraint type when a library or datasheet is output.

Options

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

Specifies a list of cell names on which the `set_constraint` command should be applied. Default: none

Note: This option currently applies only to the `-index1_factor` and `-index2_factor` options.

`-index1_factor <`*`value`*`>`

Specifies the multiplication factor. Default: `0.0`

`-index2_factor  <`*`value`*`>`

Specifies the multiplication factor. Default: `0.0`

`-margin <`*`value`*`>` Specifies the margin to add to timing constraints. Default: `0`

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

`-max_margin <`*`value`*`>`

Specifies the maximum margin allowed for the given constraint. Default: `1e20`

`-max_risefall` Specifies to use the maximum value from the `rise_constraint` and `fall_constraint`.

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

`-min_margin <`*`value`*`>`

Specifies the minimum margin allowed for the given constraint. Default: `-1e20`

`-min_recrem <`*`value`*`>`

Specifies a minimum value (in seconds) that the respective sum of recovery and removal must exceed. Default: no checking

If the minimum is not met, the removal values are adjusted as required to meet the minimum. When the `-min_recrem` value is not met, the warning level can be controlled by setting `-min_warning` as follows:

- `0` = no warnings
- `1` = 1 warning per table
- `2` = 1 warning per table value

`-min_setuphold <value>`

Specifies a minimum value (in seconds) that the respective sum of setup + hold must exceed. If the minimum is not met, the hold values are adjusted as required to meet the minimum. When the `-min_setuphold` value is not met, the warning level can be controlled by setting `-min_warning` as follows:

- `0` = no warnings
- `1` = 1 warning per table
- `2` = 1 warning per table value

`-min_warning <0 | 1 | 2>`

Specifies the warning level when `-min_recrem` or `-min_setuphold` are exceeded. Default: `1`

`-non_seq_sum_swap_direction`

Instructs the tool to sum `non_seq_setup`/`non_seq_hold` using the *opposite* direction to that used for summing setup/hold.

This only applies when the `-min_setuphold` option is set. The default is to follow the same direction as used for setup/hold, as specified by the `-sum_same_direction` option.

`-pin_dir <rise | fall | both>`

Specifies whether the margin is to be applied to just `rise` constraints, `fall` constraints, or `both` rising and falling. Default: `both`

`-pin_probe_factor`

Specifies the factors that are applied to the final margin based on the measurement of the pin to probe delay. Default: `0`

This option is applied only when `define_arc -pin_probe` is specified for the arc. The final margin for a given constraint is as follows:

```
Total_Margin = margin +  (index1_factor * pin_slew) +
(index2_factor * related_pin_slew) +  (pin_probe_factor *
pin_to_probe_path_delay) + (related_probe_factor *
related_pin_to_probe_delay)
```

`-pins {list}` Specifies a list of pin names on which the `set_constraint` command should be applied. Default: All pins

This option also accepts the asterisk (`*`) wildcard character, as shown below:

```
set_constraint -pins addr* # All pins beginning with "addr", such as addr[3]
set_constraint -pins * # All pins
```

Note: This option currently applies to only the `-index1_factor` and `-index2_factor` options.

`-related {list}` List of related pins. This option also accepts the asterisk (`*`) wildcard character.

`-rel_margin <value>`

Margin ratio to add to timing constraints. Default: `0`

`-related_probe_factor`

Specifies the factors that are applied to the final margin based on the measurement of the related pin to probe delay. Default: `0`

This option is applied only when `define_arc -related_probe` is specified for the arc. The final margin for a given constraint is as follows:

```
Total_Margin = margin +  (index1_factor * pin_slew) + (index2_factor * related_pin_slew) +  (pin_probe_factor * pin_to_probe_path_delay) + (related_probe_factor * related_pin_to_probe_delay)
```

`-sum_both_directions`

When set, all four constraint combinations (that is, `setup rise hold rise`, `setup fall hold rise`, `setup rise hold fall`, and `setup fall hold fall`) are checked for a negative sum. This is equivalent to using both settings of `-sum_same_direction` together. Default: `off`

`-sum_same_direction` Request summing using the same direction. Default: opposite direction.

The `-sum_same_direction` option can be used to check the `-min_setuphold` and `-min_recrem` options. Specify this option to use the same transition for checking the minimum sum of setup + recovery with hold + removal. The default is false, that is, sum rise + fall, fall + rise.

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

Specifies the timing constraint type.

When the `-type` option is set to `setup` or `hold`, it enables a factor of the `index_1` and `index_2` transition values to be added to the characterized constraint values. The `index1_factor` is applied to the `index_1` values and the `index2_factor` is applied to the `index_2` values. The formula is:

```
new_constr_val = orig_constr_val
    + ( index_1 * index1_factor )
    + ( index_2 * index2_factor )
    + margin
```

`-when <"function">` Logic state of side inputs.

The `set_constraint` command adds the specified `-margin` (in seconds) to the constraint type when a library or datasheet is output. Acceptable constraint types are: `setup`, `hold`, `recovery`, `removal`, and `mpw` (minimum pulse width). The margin is applied globally. If there are multiple `set_constraint` commands specified, the margin in the last command will override earlier ones. The margins are not cumulative. If no `-type` is given, the `-margin` is applied to all constraints. The `-margin` is applied before checking the value against the min/max limits. The default `-margin` is `0.0` seconds. The `-margin` is based on library thresholds and not on measurement thresholds. The `-rel_margin` option can be used to specify a relative margin to be added. The `-rel_margin` accepts a ratio number between 0 and 1.

To have an effect on the output library, this command must be executed before write_library. If the `-pin_probe` and `-related_probe` options are used with the define_arc command, and a margin is desired for the delay measurements that will be reported in the LDB, this command must be executed before char_library.

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

Allows setting of constraint-related parameters with different values either globally or for different constraint arcs without having to specify the define_arc command for each constraint.

Options

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

Specifies a list of cells that the specified criteria will affect. Default: *all cells*

If the `-cells` list is empty or not included, the specified criteria are applied globally.

`-delay_degrade <`*`value`*`>`

Specifies the maximum amount of delay degradation relative tolerance ratio permitted in the clock-to-constraint output pin (flip-flop) or the data-to-constraint output pin (latch) delay before an arriving signal is deemed to fail a timing constraint. Default: `0.1` (10%)

The *delay_degrade* is a percentage specified as a positive decimal (usually from 0.0 to 1.0). This option overrides the global parameter constraint_delay_degrade when the `-cells` option is not used.

`-delay_degrade_abstol <`*`value`*`>`

Specifies the minimum delay degradation (minimum) absolute tolerance value permitted (in seconds) in the clock-to-constraint output pin (flip-flop) or the data-to-constraint_output_pin (latch) delay. Default: `5e-12` (5 ps)

This option overrides the global parameter constraint_delay_degrade_abstol when the `-cells` option is not used. A constraint bisection search bound is deemed as failing if the delay degredation is greater than the maximum of the constraint_delay_degrade percentage of the clock-to-output-delay or data-to-output-delay and the constraint_delay_degrade_abstol exceeds the constraint_delay_degrade_abstol_max.

`-delay_degrade_abstol_max <value>`

Specifies the maximum delay degradation absolute tolerance, in seconds. Default: `-1` (off)

The `-delay_degrade_abstol` and `-delay_degrade_abstol_max` options when specified together set both an upper and a lower bound to the delay degradation. This option override the global parameter constraint_delay_degrade_abstol_max when the `-cells` option is not used.

`-delay_degrade_mode <pushout | pullin | pullin_pushout>`

Describes the degradation behavior expected on the probe mode if the metric is delay-degrade.
Default: `pushout`

Note: This option can be set on per-arc basis.

- `pushout`: Expect a normal increase in delay-to-probe as constraint approaches failure.
- `pullin`: Expect a decrease in delay-to-probe as constraint approaches failure.
- `pullin_pushout`: Expect a increase or decrease in delay-to-probe, or both, as the constraint approaches failure.

Tip

Alternatively, you set the constraint_delay_degrade_mode parameter.

`-dependent` The constraint is computed as a dependent constraint with respect to its complementary constraint (setup versus hold, recovery versus removal). This requires that constraint_dependent_setuphold should be equal to `0`.

`-dependent_margin` The absolute margin (in seconds) to be added to the independent constraint when computing the dependent constraint to which the command applies. See constraint_dependent_setuphold_margin.

`-dependent_margin_ratio`

The margin (as a multiple of the slew index) to be added to the independent constraint when computing the dependent constraint to which the command applies. See constraint_dependent_setuphold_margin_ratio.

`-glitch_peak <value>`

Specifies the maximum size of logic glitch peak relative tolerance ratio permitted on the probe node before an arriving signal is deemed to fail a timing constraint. Default: `0.1`

This option overrides the global parameter constraint_glitch_peak when the `-cells` option is not used.

`-independent` The constraint is computed as an independent constraint. This option is used to override prior `-dependent` settings.

`-metric <list>` Sets the timing constraint measurement criteria. The supported values are: `delay`, `delay_both_edges`, `glitch`, `slew`, `width`, or `path_delta`.

If the `-metric` option 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* algorithm decision process applies. The `delay_both_edges` metric applies to constraints where the probe node transitions in response to both the related and the constrained pins. The constraint is tested with delays from both the constrained and related pins to the associated transition on the probed node. Degradation failure on either edge indicates a failure of the constraint.

`-min_constraint {cell pin ...}`

Instructs Liberate to put in the library the minimum constraint (instead of the maximum constraint) for the specified cell-pin pairs. This command option accepts a list of "`cell pin`" pairs where "`cell`" specifies the name of the cell (an asterisk "`*`" indicates all cells) and "`pin`" specifies the name of a pin in the specified cell.

`-output_load <min | max | value | index_?>`

Specifies the load to be applied to cell output pins during constraint characterization. The value must be in Farads. Default: `min` (from `constraint_output_load`)

This option can be used, for example, to remove glitch suppression for cells with an expected output glitch from the clock when a reset or set pin is active. This option overrides the global parameter constraint_output_load when the cells option is not used.

`-pin {`*`pins`*`}` Specifies a list of pins. Default: `{}` (no pins)

For a constraint, the pin is constrained so that it must arrive before (setup), holds after, or meets the minimum pulse width of the characterized value. That is, the “pin” is constrained.

`-pin_dir <R | F>` Specifies the switching direction of the pin.

`-probe {`*`probes`*`}` Takes a list of nodes where one or more constraint criteria can be specified after any probe in the list. For related details, see Using the -probe option.

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

Specifies a list of related_pin(s) for the constraint. The related pin is typically a clock pin.

`-related_pin_dir <R | F>`

Specifies the switching direction of the related_pin.

`-slew_degrade <`*`value`*`>`

Includes slew degradation relative tolerance ratio when determining timing constraints. Default: `-1` (not enabled))

By setting the slew criteria, both the delay and slew degradations are checked and the first criteria to fail determines the setup and hold values. The slew degradation is a value (`0.0` to `1.0`) that represents the percentage of slew degradation and is measured using the `measure_ slew_*` parameters. This option overrides the global parameter constraint_slew_degrade when the cells option is not used.

`-type <hold | mpw | min_pulse_width | nochange | non_seq_hold | non_seq_setup | recovery | removal | setup>`

Specifies that type of constraint to which the criteria applies.

Specify the type `nochange` to apply the criteria to any of the `nochange_*_*` arc types. Use the options `-pin_dir` and `-related_pin_dir` to apply the criteria to a specific nochange arc. The `define_arc` command must be specified to enable `nochange` arcs to be modeled.

Note: This option is ignored if the `-cells` option is not specified.

`-when <"function">` Defines the logic conditions of the side and non-switching pins of the cell. It corresponds to the Liberty `-when` option.

`-width_degrade <value>`

Specifies the percentage of degradation in the width of a pulse when calculating setup/hold time at the output or internal node of the pulse generator in pulse latch cells. Default: follows the constraint_width_degrade parameter.

This is used together with the `-metric width` option of the define_arc command. This option overrides the global parameter constraint_width_degrade when the cells option is not used.

To set the global constraint-related criteria using the `set_constraint_criteria` command, do not include any of the arc-specific settings such as the options `-cells`, `-type`, `-pin`, `-pin_dir`, `-related_pin`, and `-related_pin_dir`. These options (`cells`, `-type`, `-pin`, `-pin_dir`, `-related_pin`, and `-related_pin_dir`) are used to identify specific constraint arcs to which the command will apply and can be used in any combination with any or all of the remaining options. Wildcard expressions (* and ?) may be used with `-cells`, `-pin`, and `-related_pin`. Multiple `set_constraint_criteria` commands can be applied to a single constraint. If the same option (such as `-glitch_peak`) is supplied by more than one `set_constraint_criteria` command, the last value defined is used. To set a global parameter, the `cells`, `-type`, `-pin`, `-pin_dir`, `-related_pin`, and `-related_pin_dir` should not be used.

This command can be used to set the related global parameters, but if both the global parameter and the `set_constraint_criteria` (in global mode) commands are used, then the last one executed sets the global criteria.

If a constraint criteria is set globally, using `set_constraint_criteria` and using `define_arc -metric_thresh`, the order of precedence is as follows:

- 1st: define_arc
- 2nd: set_constraint_criteria
- 3rd: global parameter

The `set_constraint_criteria` command can be used to override metric thresh values, but it does not control the metric used to measure a constraint. Use the `-metric` and `-metric_thresh` options of the `define_arc` command to control the metric to be used to measure a constraint. For example if `-glitch_peak` is supplied, it does not require that the matching constraint use a glitch measurement, but if the glitch metric is chosen by Liberate for the arc, the specified value is used in place of the global parameter value.

This command must be used before char_library.

Using the -probe option

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 > ... }

For example:

```
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.

Specifying probe nodes

To specify probe nodes to use on an arc basis, you may combine the `-related_pin`, `-related_pin_dir`, `-pin`, `-pin_dir`, `-probe`, `-cell`, and `-type` options. When specifying a probe, no other options can be used with the `set_constraint_criteria` command. That is, the probe(s) must be specified in a separate command from the other constraint criteria. For example, the criteria such as delay_degrade cannot be combined in a command with the `-probe` option. Separate commands may be used: one with the failure criteria, and one with the probe criteria.

Examples

Example 1

```
# 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
```

Example 2

```
# Specify a probe node for setup of D to CK
```

```
set_constraint_criteria \
  -type setup \
  -pin D \
  -related_pin CK \
  -probe N1 \
  -cells { dff1 dff2 }

# Specify the delay pushout failure criteria for setup D to CK
set_constraint_criteria \
  -type setup \
  -delay_degrade 0.08 \
  -cells { dff1 dff2 }
```

set_context

Integrates the derate parameters from a layout parameter extract (LPE) netlist file to the specified arcs, cells, and types during characterization.

Options

`-bin <string>`

Specifies the parameter group's ID.

`-params {<list>}`

Specifies a list of parameter names and corresponding values to define a parameter group. For example, `{param1=0.8 param2=0.9}`

`-type [ combinational | constraint | edge | async | enable | disable | retain | power |hidden | setup | hold | recovery | removal | non_seq_setup | non_seq_hold | nochange_low_low | nochange_low_high | nochange_high_low |nochange_high_high | mpw ]`

Specifies the supported arc types. Default: `combinational`

`-when <string>`

Specifies the state dependency. Default: `""` (do not apply a state).

`-pin {list}`

Specifies a list of destination pins for the arc (typically, output pins for combinational arcs, input pins for timing constraint).

`-pin_dir <string>`

The direction of the pins. The supported values are `r`, `R`, `f`, and `F`. Default: `"*"`

`-related_pin {list}`

List of input or related pins. Default: `{*}` (all pins)

`-related_pin_dir <string>`

The direction of the related pins. The supported values are: `r`, `R`, `f`, `F`. Default: `"*"`

`-probe {list}`

List of monitor node names.

`-points {list}`

List of points that need to be characterized. For example, `{{3 3} {4 4}}` or `{{1} {2}}` for MPW

`-cells {list}`

List of cell names.

`<binName>`

Specifies the name of the parameter group.

This command must be set before the char_library command. Also, ensure that this command is used along with `set_var lpe_derate_mode 1`.

Example

```
foreach cell $cells {
    set_context -bin 1 -type {list of types} -params {list of params and values}
    -cells $cells BIN_1
    set_context -bin 2 -type {list of types} -params {list of params and values}
    -cells $cells BIN_2
}
```

set_default_group

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

Options

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

Specifies a list of cell names to which this command should apply. Default: *all cells* (the command applies to all cells globally)

`-criteria {<delay | constraint | power | leakage | cap> <off | force_off | min | avg | max | min_max>}`

Specifies the method for selecting the values in the default group data tables. Supported values are different depending on the type of group. This option accepts a list of pairs of type and value. Default: `{}` (use `max` for all criteria except leakage that uses `avg`)

This option replaces the `default_timing` parameter and, if used globally, overwrites the value of that parameter. The available types, values and their default values are:

Type	Values	Default
delay	off | force_off | min | max	max
constraint	off | max	max
power	off | min | avg | max	max
leakage	off | min | avg | max	avg
cap	min | avg | max	max

The value of `force_off` tells Liberate to remove one of the rise/fall delay groups that already have state-dependent groups from the default timing group. This value is supported in the global mode only. That is, the `force_off` value cannot be used with the `-cells` option.

To select both `min` and `max` default delay groups from all `when` conditions for an arc, set the `set_default_group -criteria {delay min_max}` command. The `min` default group is marked with a user-defined attribute `"min_delay_table : true"`.

`-method {<default | const> <bitwise | table>}`

List of matched pairs. Default: `table`

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

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

`-related_pin {`*`list`*`}`

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

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

`-unateness <merge | separate>`

Keeps `positive_unate` and `negative_unate` timing groups from being merged in the default group. Default: `merge` (if both `positive_unate` and `negative_unate` timing groups exist, they will be merged into a single `non_unate` default group)

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

Note: If all `timing_sense` attributes are identical, the original timing sense remains unchanged during the merge operation. This option replaces the `default_unateness` parameter and, if used globally, will overwrite the value of that parameter.

`-when <`*`string`*`>` 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: `-when`, `-type`, `-pin`, `-pin_dir`, `-related_pin`, and `-cells` that correspond to one characterized arc for the list of cells. The `-when` and `-type` options are required. The other options are optional since they have reasonable defaults. The `-cell`, `-pin`, and `-related_pin` accept a wildcard. For example:

```
set_default_group -type leakage -when "!(A1)" -cells "XOR3D1BWP"
set_default_group -type delay -pin "Z" -related_pin "A" -when "A2 & A3" \
```

This command must be used before char_library.

Important Points to Notes:

- 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.
- The parameters that are replaced by this command will still be used internally when no list of cells is provided and, if currently accessible, would still be accessible through Tcl. At some time in the future, these parameters will not be supported. We strongly recommend this command be utilized *instead* of the global parameters.

- 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.
- The recommended settings for Liberate are to use default groups for NLDM models, but *not* CCS or ECSM models.

Using the -method option

The `-method` option specifies the algorithm for selecting the data when creating the default group. With this option, you can specify a unique selection criterion for const (constraints) and default (all other data groups). The default setting for '`default`' is `table`; the default setting for `const` is `bitwise`.

When constructing a default group using the table method, Liberate will find the worst value (min/max) from all the relevant tables and select the table that contains that value, or if avg is requested, then the table with the greatest avg value is selected. When the method is `bitwise`, Liberate will construct a default group by selecting the worst value (min/max) from all the relevant tables, on a bitwise basis or for avg, the average of the values from each table on a bitwise basis.

This option allows the user to specify the method to be applied to the timing (delay/transition), power, and cap tables (see default), and to the constraint tables (see const). The transition table in the default group will follow the timing table selection. This option requires a list of paired values. Each pair consists of a group type and a method. Acceptable group types are `default` and `const`. Acceptable selection methods are `bitwise` and `table`. This option replaces the default_group_method parameter and, if used globally, will overwrite the value of that parameter. There are 2 algorithms available when requesting an avg default power group. For more information, see the default_power_avg_mode parameter.

Example

```
set_default_group \
    -method { default table const table } \
    -unateness merge \
    -criteria { delay max power avg leakage avg cap avg }\
    -cells { inv nr2 }
```

set_dependent_load

Specifies a load to be added 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.

The dependent load is used for all characterizations such as *delay, power, constraint*, *min_pulse_width* and so on.

The default is to use the same load as the active output for input-to-output arcs. For hidden power arcs, since there is no 'observed' output port, all the output ports will be given the dependent load.

Options

`-cells {cell_names}` List of cell names. Default: all cells

`-pinlist {pin_names}`

List of pin names. Default: all pins

`<min | max | load_value | index_?>`

Default: Apply same load as arc output pin min is the minimum delay load index value.

`max`	Maximum delay load index value.
`load_value`	value of load to add to dependent pins, in Farads
`index_?`	specifies the load index value to use from load index on the delay table. It requires an integer starting with 0. Therefore, index_0 would be the first index.

This command must be used before char_library.

Example

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

set_driver_cell

Defines an active pre-driver cell to be used instead of the linear waveform. By default, Liberate uses a linear ramp as the input waveform during characterization.

Options

`-accuracy_mode < 0 | 1 >`

Enables an algorithm for generating normalized waveforms while observing the slew behavior. Default: `1`

`-char_pin <pin>` Primary output pin.

`-input_transition <value>`

Input transition time, in seconds. Default: `5e-12`

`-input_use_index` Forces the input transition of the driver to use the index from the characterized cells. This option overrides (ignores) the `-input_transition` option.

`-instantiate` Instantiates the driver cell in the simulation decks. For related details, see Using the -instantiate option.

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

Specifies a list of `driver_cell_pin` to `char_cell_pin` pairs. This option defines to which cell the pin pairs in the `-pinlist` are mapped. The `-pin_map` maps by position to the `-pinlist` pins with the first `-pin_map` cell corresponding to the first `-pinlist` pair, and so on.

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

Specifies a list of pin pairs between driver cell output pins and characterization cell input pins.

If a `-pinlist` is given, the driver cell is used for only the specific cell and pin pairs in the `-pinlist`. The wildcard asterisk (*) can be used to specify the cell names.

`<driver_cell>` Specifies the name of driver cell.

The pre-driver cell that the `set_driver_cell` command defines is driven to its 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 are equivalent to the input transitions specified in the **define_template** or define_index commands when measured at the measure_slew* voltage levels.

Note: All driver cells must be scheduled for characterization by char_library or they will be ignored. To schedule a cell for characterization, add the cell to the char_library -cells list, or do not use the -cells option at all. If the -cells option is not used, then all cells with a define_cell command and a loaded netlist (see read_spice) will be scheduled for characterization. Also, if you do not want a driver cell to be modeled in an output library, then the define_cell command for the driver cell should *not* refer to any templates (see define_template).

While characterizing NLDM delay models, you can use this command for defining the input waveform shape. For detailed information, see Delay Models in Chapter 7, "Performing Characterization using Liberate."

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

Note: The predriver_waveform will override the active driver for the related_pin in an arc. Also, the active driver can be used to drive the side pins (See the -instantiate option) while the predriver_waveform drives the related_pin.

Currently, there is no limit to the number of set_driver_cell commands you can specify.

This command must be used before char_library.

Using the -instantiate option

The -instantiate option allows the specified driver cell to be used to drive the specified cell/pin in the SPICE deck when the pin is a side pin and is static. By default, the driver instantiation will only be applied to side input pins that connect to a transistor channel. To apply a driver cell to the gate input of a side pin, the parameter driver_cell_all_inputs must be set to 1. The use of this functionality can result in a significant (typically > 20%) run time penalty due to the increase of the size of the simulation deck. (See parameter driver_cell_all_inputs.)

Liberate 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. If the driver cell has more than 1 output pin, use the -char_pin option to specify the primary output pin where slew matching is performed. The transition will be measured on the char_pin. If the -char_pin option is specified, the -pin_map and -pinlist options are also required.

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_driver_waveforms_file

Specifies a filename that holds the driver waveforms computed on the server side (see set_driver_cell). This command is used with the packet mode (see Chapter 3, "Parallel Processing,") where the clients can reuse the waveforms stored in the `driver_waveform_file` without having to regenerate them.

Note: This command is now supported in both cell and arc packet flow (see `packet_mode`).

Options

`-slew_select <all | hybrid | index>`

Specifies which slews to store for reuse by the clients. Default: `all`

The supported settings are:

- `all`: select the superset of all slews as specified in the `define_template` and `define_index` commands. (Default)
- `hybrid`: select all the slews that map to the user-specified `define_arc` command. This mode assumes that the *Inside View* algorithm is not augmenting the characterized arcs and that all characterized arcs have associated `define_arc` commands (see `char_library -user_arcs_only` and `-io`).
- `index`: select the slews specified by all the `define_index` commands.

`-mpvt_driver_dir <true | false>`

This option is specifically provided for multi-PVT flow and active driver. Setting the `-mpvt_driver_dir` option to `true` enables to generate the `driver_waveform` file and to use it on a per PVT basis by creating separate directories for each PVT. When you set active driver along with setting `set_driver_waveforms_file -mpvt_driver_dir`, active driver LDBs are written for each the operating condition. The purpose of this option is to store the NDW in a way that it can be reused.

<file_name> Specifies the name of the file where driver waveforms are stored by the server in packet flow. (Required)

This command must be used before char_library.

set_em_skip_monitor

Specifies the nets to be ignored for electromigration (EM) analysis. The EM characteristics of the matching nets will not be monitored.

Options

`-cells {<cell_names>}`

Specifies a list of cells.

Note: This option supports the use of the wildcard character "*".

`-net {<list>}` Specifies a list of pins.

Note: This option supports the use of the wildcard character "*".

`-layer {<list>}` Specifies a list of layers.

Note: This option does not support use of any wildcard.

This command must be used before char_library.

Examples

```
set_em_skip_monitor \
    -cell { DFF} \
    -net {NET1 NET2} \
    -layer {M0 M1} \
```

The above example instructs Liberate to ignore the EM results of the nets `NET1` and `NET2` on process layers `M0` and `M1` while calculating the `max_toggle_rate` for any of the pins on the cell `DFF`.

```
set_em_skip_monitor \
    -net {VDD} \
    -layer {M0} \
```

The above example instructs Liberate to ignore the EM result of net `VDD` on process layer `M0` for all cells.

```
set_em_skip_monitor \
    -net {TVDD} \
    -layer {*} \
```

The above example instructs Liberate to ignore the EM result of net `TVDD` on all process layers for all cells.

set_gnd

The `set_gnd` and `set_vdd` commands identify the name(s) of ground nets and power supply nets, respectively. Liberate must know the names of the supply nets and their respective voltages. Multiple `set_gnd` and `set_vdd` commands can be specified.

Prior to LIBERATE 19.1 and the updates to the Multi-PVT (MPVT) flows that came with it, some scripts could be run without explicitly identifying all supply and ground pins. For example, some flows could omit `set_vdd` for VDD and `set_gnd` for VSS, and still get correct results. In releases after LIBERATE 19.1, all flows must explicitly define supply and ground pins, including VDD and VSS.

Note: A warning is generated if `set_gnd` sets a pin to large positive voltage or `set_vdd` sets a pin to 0 volts.

Options

`-attributes {`*`name value ... `*`}`

List of attributes specified as name-value pairs to include in the `pg_pin` group. Default: none

For example:

```
-attributes {leakage_bin 1.1 power_negative allow}
```

Note: The preferred method of adding these types of attributes is through `write_library` with the `-user_data` option. For more information, see the description of the write_library command and the user_data_override parameter.

`-cells`

List of cells which will use this supply specification. When the `-cells` option is used, the `-name_map` option should also be used. If the `-cells` option is specified without using the `-name_map` option, the supply name will default to "`gnd_altos_<`*`gnd_value`*`>`", where the *`gnd_value`* is the voltage value of the supply with the period symbol ("**.**") replaced by the underscore ("_").

`-combine_rail`

Combines the power and leakage from the supplies specified in the `-include` option with the supply being declared by the `set_gnd` or `set_vdd` command. The power in the combined rail cannot be ignored. The command that specifies the combined power must be executed after all other `set_gnd` and `set_vdd` commands.

`-ignore_power`

Ignore the power contribution from this supply net. That is, the current in the specified supply net will not be summed into any power measurement.

Note: The `ignore_power` option will be skipped if the `-cells` option is specified.

`-include { `*`list`*` }`

List of power rails to be merged into the power rail specified in the `set_gnd` or `set_vdd` command. Default: `{}`

Note: To merge the power, the `-combine_rail` option must also be specified.

`-name_map <value>`	Specifies the name that this supply will be called in the `output .lib` file. Name mapping is only supported when the `pg_pin` (see the pin_based_power parameter) syntax is enabled.
	
	The `-cells` option is often used with the `-name_map` option to change the `pg_pin` name on an individual cell basis. This allows the mapping of a global supply to a local cell-specific supply possibly at a different voltage. For example:
	`set_vdd vdd`
	`set_vdd -cells {INV_X1}  -name_map VDD_X1  VDD 0.9`
	`set_vdd -cells {INV_X16}  -name_map VDD_X16  VDD 0.8`

`-no_model`	Request to not include this supply in the output .lib.
`-type <primary | backup | internal | nwell | pwell | deepnwell or deeppwell>`	
	Specify the power supply type. (Default: `primary`)
`<net_name>`	Name of ground/power supply net.
`<voltage_value>`	Voltage value (in Volts).

Liberate automatically identifies `0`, `GND`, and `VSS` (case-insensitive) as ground supplies and sets them to zero volts. Use the `set_gnd` command to set them to alternative values. It is not recommended to attempt to change the voltage of the ground net `0` because this is considered the reference ground.

Liberate automatically identifies `VDD` (case-insensitive) as power supplies and set them to the default voltage specified by the set_operating_condition command. Use the `set_vdd` command to set them to alternative values.

Note: Liberate generates a warning message if `set_vdd` is used to set a pin to `0` volts.

This command should be used before the read_spice command to have the desired effect.

This command must be used before the char_library command.

Examples

Example 1

```
set_vdd -combine_rail -cells {test1 test2} -include {VDD} VDD33 1.1
```

In the above example, the contributions for power and leakage from rail VDD will be merged to `VDD33` for cell `test1` and `test2`. The VDD will appear in the library as defined power rail only (that is, no power and leakage values is specified for VDD) if `-no_model` is not specified for VDD in `test1` and `test2` in other `set_vdd` command.

```
# Set VDD3 to 3 volts
set_vdd VDD3 3
set_gnd BULK_GND 0
```

Example 2

```
set_vdd -cells {LVLSHIFT} -name_map COREVDD1 \
    -attribute {std_cell_main_rail true} VDD 0.72
set_gnd -cells {LVLSHIFT} -name_map COREGND1 -type primary VSS 0.0
set_operating_condition -voltage 0.72 -temp 125 -name "$pvt"
set_pin_vdd -leakage_add_to_supply VDD \
    -supply_name COREVDD2 LVLSHIFT I 0.81
set_pin_vdd -supply_name COREVDD1  LVLSHIFT Z 0.72
```

The above example shows how to configure a level-shifter cell.

The `voltage_name` and `output_signal_level` for pin `Z` is written as `COREVDD1`.

The `voltage_name` and `input_signal_level` for pin `I` is written as `COREVDD2`.

set_input_voltage

Creates `input_voltage` groups at the library and pin level. For a default group the naming convention is: `(default_$vdd_$gnd_$direction)`. For example:

```
input_voltage(default_VDD1_VSS1_output)
```

For multiple groups, the naming convention is: `(user_voltage_$num)`.

Options

`-vil <value>`	Specifies voltage input low.
`-vih <value>`	Specifies voltage input high.
`-vimin <value>`	Specifies minimum input voltage.
`-vimax <value>`	Specifies max input voltage.
`-cells {list}`	Specifies a list of cells.
`-pins {list}`	Specifies a list of pins.
`-name "string"`	Specifies the name of the `input_voltage` group. Default: Liberate determines the group name.

Example

```
set_input_voltage  \
 -vil 0 -vih 1.2 \
 -vimin 0 -vimax 1.2 \
 -name default_in

  input_voltage (default_in) {
    vil : 0;
    vih : 1.2;
    vimin : 0;
    vimax : 1.2;
  }
```

set_ldb_comment

Adds comments to the library database (LDB).

Syntax

```
set_ldb_comment <tcl_format_string>
```

Options

`<tcl_format_string>` Specifies a string in Tcl format.

This command should be used before the char_library command is run.

Example

In the Tcl file:

```
set str [string cat \
    "ldb_header_simulator_version : \"[get_var extsim_cmd]\"; \n" \
    "ldb_header_date : \"[exec date]\"; \n" \
    "ldb_header_version : \"1.0pre\";" \
]
set_ldb_comment $str
```

In the LDB:

```
altos_compile_version : "21.1.0.999";
altos_compile_date : "Fri Sep 18 09:36:28 PDT 2020";
/*
ldb_header_simulator_version : "";
ldb_header_date : "Fri Sep 18 09:48:55 PDT 2020";
ldb_header_version : "1.0pre";
*/
revision : "1.0";
comment : "";
```

set_logic_condition

Specifies a logic condition to insert into an arc to reduce/filter potential vectors and thereby reduce the simulation overhead.

Options

`-cells {`*`list`*`}`	List of cell names. Default: `"*"` (all cells)
`-pin {`*`list`*`}`	List of pin names. Default: `"*"` (all pins)
`-pin_dir [R | F]`	The pin direction. Default: `""` (apply to all directions)
`-related_pin {`*`list`*`}`	List of related pin names. Default: `"*"` (all pins)
`-related_pin_dir string ()`	The related pin direction [R|F]. Default: `""` (apply to all directions)
`-type {`*`list`*`}`	The type of the arc. Supported types are constraint, delay, hidden, leakage, mpw, setup, and hold. Default: `""` (Apply to all types)
<logic_condition>	The logic condition to be applied to filter vectors.

This command adds and applies the specified `logic_condition` to all matching arcs. Multiple `set_logic_condition` commands can be used. However, if more than one command maps to a particular arc, the one that most closely matches the arc or the last one specified is applied. This command works in the same way as the `-logic_condition` option of the define_arc command and can be used to augment any `define_arc` commands that overlap with the option settings of this command without having to edit the actual `define_arc` command.

Note: For details about handling logical conflicts between this command and the `define_arc -when` command, see set_logic_condition_resolve_conflicts.

This command must be used before char_library.

Example

```
set_logic_condition \
    -type delay \
    -pin "Z" \
```

```
-related_pin "A1" \
-cells { ao32 } \
"B1 & C"
```

set_max_fanout

Liberate can compute the *max_fanout* attribute for all output and bidi pins. The value is computed as follows:

```
max_fanout = <max_capacitance> / <input cap for refcell/refcell_pin>
```

Where:

```
<max_capacitance> == max_capacitance attribute value for an output or bidi pin
for the cell


<input cap for refcell/refcell_pin>  == the input cap of chosen pin for the
cell.
```

Options

`-refcell <value>` The reference cell.

`-refcell_pin <value>`
The pin of the reference cell.

`-refcell_dir <Rise | Fall>`
The direction of the reference pin.

`-cells {list}` List of cells.

set_message

Limits the output of the selected messages based on the message ID and severity.

Note: This command works only for those messages that have a unique message ID.

Options

`-id { list }`
Specify a list of message IDs. The ID can be specified as `966` or `Lib-966`.

Note: When the ID is specified in the `Lib-<ID>` format, only the numeric part of the ID is used.

`-severity < info | warn | error >`
Apply the `set_message` command to only the specified type of messages.

`-msg_limit < -1 | 0 | value>0 >`
Limit the number of messages to print.

- `-1` (default): Instructs to follow the setting of the msg_limit_per_type_per_cell parameter.
- `0`: Instructs to not print the messages.
- `value`: Specifies the number of matching messages to write out.

This command can be specified multiple times in a run, but ensure that it is specified before the char_library command is run.

set_network_port

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

Options

`<port_number>`	Specifies the network port number.

By default, Liberate will search for an available port. To specify the machines to use for parallel processing use multiple `set_client` commands. For more details, see the Distributed Processing section in Chapter 3, "Parallel Processing."

Note: The TCP/IP port is determined by the following equation, if it has not been specified with the set_network_port command:

```
30003 + rand() % 1413
```

and falls within 300003-30259, 30261-30998, 31000-31019, 31021-31415.

This command must be used before char_library.

Example

```
# 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/liberate linux1
set_client -dir /tmp/liberate linux2
```

set_operating_condition

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

Note: The `set_operating_condition` command will not override any supply voltage that was previously specified using the set_vdd command.

Options

`-name <value>` Specifies the operating condition name to be used in the output library.

`-no_model_default_supplies <false | true>`

Instructs Liberate to create the default supplies (VDD, VSS, GND, and 0) without modeling them. Default: 0 or false

`-supply_name <string>`

Specifies the operating condition name and instructs Liberate to use the voltage set by the `set_vdd` command for the `supply_name` as the default operating condition voltage in the generated `.lib` file.

If this option is not specified, Liberate, by default, uses the `-voltage` and `-temp` options in a constructed name as the default operating condition in the generated `.lib` file.

When `-supply_name` is used, the `operating_condition -voltage` command option will not override the voltage set by the `set_vdd` command.

Note: The `set_vdd` command should be specified before the `set_operating_condition` command.

`-temp <value>` Defines the temperature to characterize in °Celsius. (REQUIRED)

`-voltage <value>` Specifies the default positive supply voltage in volts. Default: `0V` (REQUIRED)

This voltage will be assigned to any VDD pin name. To specify additional power or ground supply nets and their appropriate values use the **set_gnd** commands. The default supply names are VDD for the positive supply, VSS, GND, and 0 for the negative supply.

This command must be used before the char_library command.

Examples

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

```
    -voltage 1.2
```

If the supply_info parameter is set to `1` along with the `set_operating_condition` command, messages such as shown below will be displayed.

```
set_operating_condition -voltage 3.0 -temp 100 -no_model_default_supplies


Vdd/gnd summary: (set_vdd): Pin 'VDD' is set to 3 V. -no_model 'True' -ignore_power
'False' -type 'primary' -attributes {} -combine_rail 'False' -include {}
Vdd/gnd summary: (set_gnd): Pin 'VSS' is set to 0 V. -no_model 'True' -ignore_power
'False' -type 'primary' -attributes {} -combine_rail 'False' -include {}
Vdd/gnd summary: (set_gnd): Pin '0' is set to 0 V. -no_model 'True' -ignore_power
'False' -type 'primary' -attributes {} -combine_rail 'False' -include {}
Vdd/gnd summary: (set_gnd): Pin 'GND' is set to 0 V. -no_model 'True' -ignore_power
'False' -type 'primary' -attributes {} -combine_rail 'False' -include {}
```

set_output_voltage

Creates `ouput_voltage` groups at the library and pin level.

Options

`-vol <value>`	Specifies voltage output low.
`-voh <value>`	Specifies voltage output high.
`-vomin <value>`	Specifies minimum output voltage.
`-vomax <value>`	Specifies max output voltage.
`-cells {list}`	Specifies a list of cells.
`-pins {list}`	Specifies a list of pins.
`-name "string"`	Specifies the name of the `output_voltage` group. Default: Liberate determines the group name.

For a default group, the naming convention is: `(default_$vdd_$gnd_$direction)`. For example:

```
output_voltage(default_VDD1_VSS1_output)
```

For multiple groups, the naming convention is: `(user_voltage_$num)`. For example:

```
    output_voltage(user_voltage_0)
    output_voltage(user_voltage_1)
    output_voltage(user_voltage_2)
```

Example

```
set_output_voltage     \
    -vol     0  -voh     1.2 \
    -vomin 0 -vomax 1.2 \
    -name default_out

output_voltage (default_out) {
    vol : 0;
    voh : 1.2;
    vomin : 0;
    vomax : 1.2;
}
```

set_packet_controls

Defines explicit maximum job weights and per-arc weighting factors to be used in the Bolt job distribution system flows. This command can be useful in tuning Bolt for different use cases.

By default, the tool will use the internal defaults for weighting.

Options

`-max_weight <integer>`

Define the maximum weight of any Bolt job.

`-weight_factors <list>`

Specify a list of weight-criteria/value pairs.

The following criteria are accepted: `delay`, `search`, `hidden`, `leakage`, `mpw`, `special_cap`, `missing_cap`, `ccsn`, `ibis_IV`, `ibis_VT`, and `min_period`.

This command must be used before the char_library command is run.

Examples

Example 1

```
# Liberate Base cells
set_packet_controls \
    -max_weight 600 \
    -weight_factors { \
        ccsn 15 \
        delay 15 \
        hidden 15 \
        leakage 1 \
        mpw 150 \
        search 150 \
    }
```

Example 2

```
# Liberate MBFF cells
set_packet_controls \
```

```
    -max_weight 150 \
    -weight_factors { \
        ccsn 150 \
        delay 30 \
        hidden 30 \
        leakage 1 \
        mpw 150 \
        search 150 \
    }
```

Example 3

```
# Liberate Variety Base cells
set_packet_controls \
    -max_weight 600 \
    -weight_factors { \
        delay 50 \
        mpw 150 \
        search 150 \
    }
```

Example 4

```
# Liberate Variety MBFF cells
set_packet_controls \
    -max_weight 150 \
    -weight_factors { \
        delay 50 \
        mpw 150 \
        search 150 \
    }
```

set_pin_attribute

Modifies pin attributes in the library.

Options

`-abstol <value>`	Absolute threshold (in seconds) is used to compare the result of `abs(cell_rise/cell_fall)`. (Required) The specified absolute threshold value must not be negative. For example, use `1e-12` for `1ps`.
`-attributes {list}`	List of attributes that will be updated. Default: `""` When the `-type` option is set to `min_max_cap_tran`, the supported attributes are: `max_transition`, `min_transition`, `max_capacitance`, and `min_capacitance`. The `-attributes` option can be set to these and if it is not set, it will default to these attributes.
`-cells {list}`	List of cells. Currently, it accepts only one cell name and does not support use of wildcards. (Required)
`-pin "string"`	Arc pin, that is, the output or constrainted pin. (Required)
`-related_pin "string"`	Arc related_pin. (Required)
`-reltol <value>`	Relative threshold that is used to compare the result of: `abs(cell_rise - cell_fall) / (cell_rise+cell_fall)`. (Required) The specified relative threshold value must not be negative, but must be less than `1`. For example, use `0.05` for `5%`.
`-type "string"`	The type of data to be modified. Currently, it accepts only `"min_max_cap_tran"`. (Required) When `-type` is specified as `min_max_cap_tran`, a maximum delay sub-table is chosen from the characterized delay table for the arc specified by the pin, related_pin, and cell. This sub-table is chosen according to the absolute threshold and relative threshold values provided. The attributes `max_transition`, `min_transition`, `max_capacitance`, and `min_capacitance` can be set to match the sub-table.
`-when "string"`	The arc conditional state, that is, the when condition.

This command can be used after char_library, read_ldb, and read_library, but it must be used before model generation such as with write_library.

You can specify multiple `set_pin_attribute` commands.

Example

```
read_ldb test.ldb.gz
set_pin_attribute -type min_max_cap_tran -abstol 4e-10 -reltol 0.8 -pin Y
-related_pin A -cells {INVX1}
write_library -filename my.lib test
```

set_pin_capacitance

Specifies how the `pin_capacitance`, `rise_capacitance`, and `fall_capacitance` attributes and the default advanced table capacitance (`ccs_receiver_capacitance_rise` and `ccs_receiver_capacitance_fall`) models for each pin are determined. By default, the NLDM `pin_capacitance` is selected from the maximum value of all the calculated capacitances without considering the direction (rise or fall), logic state, or slew/load in the characterized state-dependent capacitance tables. By default, the advanced table-based capacitance models are selected from one of the characterized capacitance tables.

Options

`-range_use_side_input`

Changes the *capacitance_range* calculation to observe the `-side_input` option setting.

This option can be used as a post-processing step, as long as the set_pin_capacitance command is invoked again after the read_ldb command. For example:

```
set_pin_capacitance -state avg -table avg \
    -direction min \
    -side_input noncontrolling -range_use_side_input
```

`-side_input <controlling | noncontrolling | all>`

Specifies the method to be used to select different groups of vectors. Default: `all`

When set to `controlling`, the pin capacitance will be measured only for vectors where there is a side input that is controlling the output. This is usually the case only for hidden vectors.

When set to `noncontrolling`, the pin capacitance will only be measured for vectors where no side input controls the output. That is, the related_pin controls the output. This will not include any hidden vectors.

The default is `all`, which means to measure pin capacitance for all vectors.

`-state <min | avg | max>`

Specifies the method to be used to select between different states. Default: `max`

`-state_avg_method <merge | separate>`

Specifies the method to be used to compute the average capacitance. Default: `merge`

For the `-state_avg_method` option to have any effect, the `set_pin_capacitance -state` command must have a value of `avg` and the `.lib` must have timing groups with multiple states merged into a single timing group. This option supports the values of `merge` (Default) and `separate`. For example, if there are two sets of *when* conditions, and `W1`, `W2`, `W3` are all logic cubes encompassing all input pins `when: W1+W2, c1` and `when: W3, c2`:

```
merge method: c_avg = (c1 + c2) / 2
separate method: c_avg = (c1 + c1 + c2) / 3
```

`-table <min | avg | max>`

Specifies the method to be used to select within a table. Default: `max`

`-direction <min | avg | max>`

Specifies the method to be used to select between rise and fall capacitance. Default: `max`

`-meas_supply_cap` Instructs Liberate to take a capacitance measurement on the supply pin. The measurement is made using the leakage deck.

Note: This option does not affect leakage characterization.

The power supply capacitance is measured by ramping down the power supply voltage. The ramping is done after leakage is measured so as to not disturb the leakage acquisition. The CCSP `parasitic_capacitance` data structures are used to store the result. The supply capacitance will be written to the output library as a commented pin group containing a capacitance attribute.

`-when <"function">` Logic conditions of side inputs.

`-vector <"vector">`	Logic conditions of side inputs that must match `define_cell -pinlist`. The `-vector` option should be consistent with the `-pin_dir` option for the ports listed in the `-pins` option. If the `-vector` option has an `"x"` for the `-pins` option, Liberate enforces consistency. If the vector has an `R` or `F`, the `-pin_dir` option must have rise or fall. There is no character supported in the `-vector` option to indicate both directions. That is, there is no support for a `"b"` in the `-vector` option.
`-pin {pin}`	Specifies the name of the pin to which the state selection criteria should be applied.
`-pin_dir <rise | fall | both>`	Specifies the pin direction.
`-cell <cell>`	Specifies the cell name.

Using this command enables numerous selection levels. For NLDM `pin_capacitance`, the first level (`table`) determines whether to use the minimum (`min`), average (`avg`), or maximum (`max`) values from each table to roll up. The next level (state) chooses the capacitance rolled up for each table in the previous step by logic state (when condition). From the capacitances selected from each table across all logic states, a single value for `rise_capacitance` and a single value for `fall_capacitance` will be selected. The final level of refinement determines how the `pin_capacitance` attribute is determined between the `rise_capacitance` and `fall_capacitance` from the previous step. The `min`, `avg`, and `max` values can be selected at each level (`table`, `state`, and `direction`). The default at all levels is `max`. This command can be used independently from characterization, only impacting the attributes output by `write_library` because the characterization database (LDB) contains all the rise and fall state-dependent capacitance tables for each pin. For the more advanced table-based models, such as the `ccs_receiver_capacitance_rise` and `ccs_receiver_capacitance_fall`, the capacitance data associated with the default `hidden_power` group will be written into the `.lib`.

This command can be used to specify a particular vector or vectors to use when measuring input pin capacitance. The following options are available to support this: `-when`, `-vector`, `-pin`, `-pin_dir`, and `-cell`. The `-pin_dir` option is required when specifying capacitance vectors. The 'vector' or the 'when' options should also be specified. If multiple vectors are desired, execute multiple `set_pin_capacitance` commands or use wildcards (x) in the vector. At least one of the `set_pin_capacitance` commands must specify satisfiable logic conditions (vectors) or the output library may not have valid pin capacitance values for the pin. In addition, the `set_pin_capacitance` command must reference an arc that is characterized; that is, the arc specified must have been characterized, matching the

when, vector, pin, pin_dir and cell. The vector option must specify the same number of pins as in the pinlist option of the corresponding define_cell command.

Example

```
# Set how the pin capacitance attributes are determined
set_pin_capacitance -state max -table avg -direction min
# specify a particular vector for the input cap of pin A on cell lag.
set_pin_capacitance -vector "x0x" -pin A -pin_dir rise -cell lag
```

set_pin_delay_threshold

Sets the `input_threshold_percent_fall` and `input_threshold_percent_rise` attributes at the pin level. This must be used together with define_arc to set the delay thresholds. This only adds the attributes into the library and does not control the characterization (which is done by `define_arc`).

Options

`-cells { list }`	List of cells. (REQUIRED)
`-pins { list }`	List of pins. (REQUIRED)
`{ values }`	Specify two delay measurement thresholds: `input_threshold_percent_rise`, `input_threshold_percent_fall`

Example

```
set_pin_delay_threshold -cells {cellA cellB} -pins {pin1 pin2} {.3 .2}
```

set_pin_gnd

See set_pin_vdd for description of options.

Options

`-add_supply`	Notifies Liberate to create a new supply with the name specified using the `-supply_name` option or an arbitrary internal supply name if `-supply_name` is not specified (and if the supply does not exist already.) **Note:** It is recommended to define all supplies using the set_gnd commands. The `-add_supply` option is available only for backward compatibility and should not be used.
`-supply_name <name>`	Name of the supply that drives this pin. (REQUIRED)
`{cell_names}`	List of cell names. (REQUIRED)
`{pin_names}`	List of pin names. For example, {a1 a2 a3 c din ckn}, or regexp like {a* c*}. (REQUIRED)
`<gnd_value>`	Ground supply value. (REQUIRED)

set_pin_slew_threshold

Lets you set the measurement thresholds for specific cell pins that need different sets of threshold. When this command is set, the thresholds will overwrite the measure threshold set by the `measure_slew_*` parameter and the `define_arc -slew_thresh` command for the specific cell(s) and pin(s).

Options

`-cells {`*`list`*`}`	List of cells (REQUIRED).
`-pins {`*`list`*`}`	Pins that need special flew threshold (REQUIRED).
{*`slew_threshold`*}	Specify the four slew measurement thresholds: `lower_rise`, `upper_rise`, `lower_fall`, and `upper_fall`. (REQUIRED)

set_pin_vdd

The `set_pin_vdd` and **set_pin_gnd** commands associate a *pin* of a cell with a particular supply domain. This is particularly useful on cells that have multiple power and ground connections, such as level shifters where it may be difficult for Liberate to correctly identify the supplies for each pin. The cell and pin options accepts a list of names. The cell and pin names may be specified with wildcard characters `*` and `?` and `regexp` expressions. In the event that multiple commands are given which map to the same cell and/or pin, then the last command given takes effect.

Options

`-add_supply` Notifies Liberate to create a new supply with the name specified using the `-supply_name` option or an arbitrary internal supply name if `-supply_name` is not specified (and if the supply does not exist already.)

Note: It is recommended to define all supplies using the set_gnd commands. The `-add_supply` option is available only for backward compatibility and should not be used.

`-leakage_add_to_supply <name>`

Specifies the name of a supply to which all leakage for this supply 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 where the input is driven by a supply domain that does not exist in the cell and you do not want to lose the input power.

`-supply_name <name>`

Specifies the name of the supply associated with the specified pin. (REQUIRED)

This option is often needed with level shifter cells to avoid Liberate matching an incorrect supply to the specified 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, It will also check to see if the input/output shares the same voltage supplies before using the arc based constructs.

When this option is used, and the voltage_map parameter is set to `1`, the specified `supply_name` will be 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_names}` List of cells. (REQUIRED)

`{pin_names}` List of pin names. For example, `{a1 a2 a3 c din ckn}`, or wildcard pattern matching like `{a* c*}`. (REQUIRED)

`<vdd_value>` Power supply value (Optional)

When you specify the `supply_name` but do not specify the `vdd_value` and the supply exists in the vdd list (see set_vdd), then the vdd voltage value from the `set_vdd` command is used. If you specify both `supply_name` and `vdd_value`, ensure that the vdd value is the same as the one supplied in the `set_vdd` command. Else, an error message is generated and the command is ignored.

1. If there are more than one set_pin_vdd commands for the same pin but with a different value or different supply name, then the second instance of the command overwrites the first.

2. If there are two supplies with different names but with the same voltage value, and a set_pin_vdd command associates a net to this voltage but the supply_name is not given, Liberate reports an error and exits (after reporting all similar errors.)

Example error cases:

```
# -supply_name not given:
set_vdd VDD1 1.2
set_vdd VDD2 1.2
set_pin_vdd NAND2 Y 1.2

# Attempt to redefine vdd2 voltage:
set_pin_vdd -supply_name vdd2 busDriver PAD 3.3
```

This command must be used before char_library.

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 $VOLT
```

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

In the above example, gate leakage currents on pin `IN` will be multiplied by `$VOLT1` (controlling `VDD1`) to get the leakage power that will be added to the supply pin `VDD`.

set_pvt

Sets the specified PVT as the active PVT. This command also executes all commands defined within the `define_pvt` command for the PVT being set.

Note: This command should be used after the last `define_pvt` command because the PVT should be defined before it can be enabled.

For more information, see Chapter 9, "Using the Multi-PVT Characterization Flow of Liberate Trio."

See also define_pvt and get_pvts.

Option

{<*pvtName*>}	Specifies a unique name of the PVT corner. (REQUIRED)

Example

```
set pvts [get_pvts]
foreach pvt $pvts {
    set_pvt $pvt
    set file_path "${rundir}/LIBS/${libprefix}_${pvt}.lib"
    puts "writing LIBFILE = $nom_file"
    write_library -filename $file_path $pvt
}
```

In the Tcl loop illustrated above, all libraries can be written with a naming convention specific to each corner.

set_receiver_cap_thresholds

Allows you to override the default behavior of Liberate with respect to `ecsm_capacitance` tables.

Options

`-rise {list}`	List of receiver rise capacitance thresholds specified as a percentage in decimal. (that is, `.5` = 50%)
`-fall {list}`	List of receiver fall capacitance thresholds specified as a percentage in decimal. (that is, `.5` = 50%)

When this command is run, Liberate characterizes the following types of thresholds:

- Superset of thresholds including those specified by the user in this command
- `measure_slew_*` thresholds
- `delay_*` thresholds
- `measure_cap*` thresholds

This means that any threshold related to capacitance measurement and modeling will be automatically added to the list of characterized thresholds.

For 1-piece tables, Liberate reports `ecsm_capacitance` at `delay_inp_rise` and `delay_inp_fall`.

For 3-piece tables, Liberate also reports `ecsm_capacitance` at `measure_slew_lower_rise`, `measure_slew_upper_rise`, `measure_slew_upper_fall`, and `measure_slew_lower_fall`. This command allows you to augment these tables with additional voltage thresholds.

Note: The receiver capacitance thresholds must be symmetric for rise and fall arcs. The first rise corresponds to last fall, the second rise corresponds to the next to last fall, and so on. Each pair should sum up to `1`. Asymmetric thresholds are not currently supported and will introduce some unwanted changes in the capacitance models.

Examples

Example 1

```
# Create ecsm_capacitance tables for 20%, 30%, %50, 70%, 80%
```

```
# for both rising and falling arcs
#
set_receiver_cap_thresholds \
    -rise { 0.2 0.3 0.5 0.7 0.8 } \
    -fall { 0.2 0.3 0.5 0.7 0.8 }
```

Example 2

```
# Compact version of Example 1, where:
# delay_*=0.5
# measure_slew_lower_*=0.3
# measure_slew_upper_*=0.7
#
set_receiver_cap_thresholds -rise { 0.2 0.8 } -fall { 0.2 0.8 }
```

Example 3

```
# Create 8 piece ecsm_capacitance tables that are weighed towards the tail of the
arc
#
set_receiver_cap_thresholds \
    -rise { 0.1 0.3 0.5 0.6 0.7 0.8 0.9 0.9999 } \
    -fall  { 0.0001 0.1 0.2 0.3 0.4 0.5 0.7 0.9 }
```

set_rsh_cmd

Defines an `rsh_cmd` string to be used with a specified list of cells when accessing a remote client through cell-based distributed parallel processing. This can be used to send a select group of cells to a specific queue (possibly one with different compute resources.)

Options

{*rsh_cmd_string*}	String to be used for the `rsh_cmd`.
{*cells*}	List of cells to which this `rsh_cmd` is applied.

Important Points to Note

- If there are more than one cell in a packet and not all the cells in that packet use the same rsh_cmd, Liberate uses the rsh_cmd associated with the cell estimated to require the most effort.
- This command is supported only in cell packet mode (see packet_mode).

This command must be used before char_library.

Example

```
# Setup an rsh_cmd for a "smallQue" machine
set_var packet_clients 10
set_var rsh_cmd "bsub -q smallQueue"

# Setup an rsh_cmd to send some big cells to a queue on a different machine
set_rsh_cmd "bsub -q bigQueue" { bigCellA bigCellB }
char_library
```

set_sim_init_condition

Instructs Liberate how to set initial conditions when presenting spice decks to external simulators, that is, it allows local/global control of `sim_init_condition`. See also **char_library** `-extsim`.

Options

`-cells { list }` List of cell names to which this command should be applied. Default: all cells

`-floating_node < init | ignore >`

Tells Liberate to initialize (default) floating nodes, or to ignore floating nodes. Default: `init`

When ignored, the external simulator can initialize the nodes or the user can provide the initialization using the `-extsim_deck_header` option with the define_arc command. When providing initial conditions with `-extsim_deck_header`, the node names must map into the deck. This requires the addition of an additional level of hierarchy of "`X1.`" to the cell nodename in the `.ic` command.

`-method < hybrid | ic | ic_except_dc | ic_except_dc_plus_leakage>`

Specifies the method for setting initial conditions. Default: `hybrid`

This is the same as the sim_init_condition parameter. Accepted values are:

- `hybrid`: Use `.ic` for any floating nodes and `.nodeset` everywhere else. (Default)
- `ic`: Always use `.ic` to initialize nodes in the external simulation.
- `ic_except_dc`: Use `.ic` to initialize nodes in all external simulations except when the measurement is a DC type measurement. This can occur for leakage measurements.
- `ic_except_dc_plus_leakage`: Same as ic_except_dc except that it forces to use `.ic` to do leakage characterization.

`-vector_skip_open_source_drain`

Tells Liberate to skip vectors where a floating low node has no pull-down or a floating high node has no pull-up. This option might be useful, for example, when there is an inverter surrounded by many decap cells. Here, the decap cell has bi-stable states - in one state both nodes are floating; in another state both nodes are connected. For example:

```
M1   VDD   N1   N2   VDD   PMOS
M2   N1   N2   VSS   VSS   NMOS
```

when `N1=0`, `N2=1`, both nodes are driven
when `N1=1`, `N2=0`, both nodes are floating, but this is an unstable state.

When the `-vector_skip_open_source_drain` option is used, the vector with `N1=1` and `N2=0` will be skipped.

This command must be used before char_library. It has multiple uses.

Examples

Example 1

```
# Request the use of .ic for all nodes needing initialization except for leakage decks (leakage uses the DC solution).
# Do not initialize floating nodes. Let the simulator handle the initialization.
set_sim_init_condition -method ic_except_dc -floating_node ignore
```

Example 2

```
# Request that the cell called myCell uses .ic to initialize all nodes
set_sim_init_condition -method ic -cells { myCell }
```

set_simultaneous_switch

Disables simultaneous switching input analysis on a cell-by-cell basis. See the simultaneous_switch parameter for more information about simultaneous switching input analysis.

Options

`-off`	Disable simultaneous_switch on the list of cells.
`{ cell_names }`	Names of cells. (REQUIRED)

Example

```
set_simultaneous_switch -off { mycell }
```

set_stress_criteria

Defines the cell-level stress options for input stimulus.

Options

`-slews`	Specifies the slews used in stress simulation. Default: First slew index in the delay template
`-loads`	Specifies the loads used in stress simulation. Default: First load index in the delay template
-duty_cycles	Specifies the duty cycle used in stress simulation. Default: Duty cycle specified using set_aging_criteria
-toggle_rate	Specifies input pins toggle rate, pair of `(<pinName>, <toggleRate)`. Default: `0.5`
-cells	Specify the cells. Default: all **Note:** You can specify the list using wildcard expressions, such as, `".*INV"`, `"INV.*"`, or `".*INV.*"`.

set_three_state

Disables three_state modeling for a list of cells. This can be used for cells such as one-hot MUXes to disable the modeling of the outputs as three-state outputs.

Options

`-current_degradation_threshold <value>`

Specifies the current degradation threshold. Default: `0.1`

The threshold specifies a ratio of the original current. When the current falls below this threshold, the output is considered off. When this option is used, all disable arcs will be characterized with a disable_method of `2` (current degradation).

`-off` Turn off three state modeling.

`{ cellList }` List of cells.

This command can be used after **char_library** except in one case. If the `-current_degradation_threshold` option is specified, this command must be used before `char_library`.

Example

```
set_three_state -current_degradation_threshold  0.01  ${cells}
```

set_units

Specifies the timing, capacitance, leakage power, and current units to be used in the output library. This command *only* affects modeling, and has no effect on either the VDB or the template files.

Units can be uppercase or lowercase; internally Liberate is case-insensitive and understands that "1mw" and "1mW" are the same. The purpose of this command is to output a library containing the correct units for your down-stream tools.

Options

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

Specify the capacitance units. Default: `1pf`

`-current < 1a | 1ma | 1ua >`

Specify the current units. Default: `1ma`

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

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

`-pulling_resistance < 1ohm | 10ohm | 100ohm | 1kohm >`

Specify the pulling resistance units. Default: `1kohm`

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

Specify the timing units. Default: `1ns`

This command must be set before any command that creates a library, such as write_library or write_verilog, and can be set after char_library.

Example

```
# Set the timing units to 1 pico second.
set_units -timing 1ps
```

set_var

Sets Liberate-specific parameters. For detailed information about the supported parameters, see Chapter 5, "Liberate Parameters."

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).
-pvt	List of PVT names for which the parameter is applicable. Use of wildcard characters is allowed.
`-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).
`-stage`	Specifies that the parameter set with the `set_var` command should be applied only during variation characterization. Default: `variation` **Note:** This option is supported only in Liberate.
`-type < constraint | delay | delay and power | hold | leakage | mpw | nochange | noise | power | recovery | removal | setup >`	Type of arc. (Default: *all types*) When `-type` is set to `noise`, the `set_var` command can set a Liberate parameter for only noise (CCSN/ECSMN) related simulation decks.
`<parameter_name>`	Parameter name (REQUIRED)
`<value>`	Parameter value (REQUIRED)

The `-cells`, `-type`, `-pin`, `-pin_dir`, `-related_pin`, and `-related_pin_dir` options are used to specify local cell and arc specific parameters. Some parameters can only be set at a global level.

Some parameters support per-cell and/or per-arc and/or per-type. All options are not valid for all parameters. If an option is not allowed, an error is issued in the log file and the `set_var` is ignored. If an option is omitted, any value for that option is allowed.

The `-cells`, `-pin`, and `-related_pin` options support the usage of the wildcards `*` and `?`.

The `set_var` commands are processed in reverse order at run time. The first value that matches the specific cell, arc, and type when processed in reverse order is applied to the characterization. It is recommended that when you are setting a global value, always ensure that the global value is set first. Then, set any local values after the global value has been set.

Example

```
set_var -cells DFF* write_logic_function false
```

set_vdd

The purpose and options of this command are similar to that of the `set_gnd` command. Therefore, see set_gnd for details on this command.

set_vtgm_cell

Specifies the cell to characterize the Threshold Voltage and Maximum Transconductance (vtgm) value.

Using this command, you can specify the cell for which the vtgm value needs to be characterized. The `model_vtgm` attribute is defined at library level. It is the maximum value of each `cell_model` attribute at cell level. The calculated vtgm value for the specified cells is placed as a cell-level attribute. The tool identifies the maximum value out of all the cell-level attribute values and places it as an attribute at the library level.

Options

`-cells { <list_of_cells> }`

Specify a list of cells.

`-pmos_sweepvol { <sweep_voltage_begin> <sweep_voltage_end> <sweep_voltage_step> <voltage_of_the_drain> <VDD> }`

Specifies the sweep input voltage information (begin, end, and step), the voltage for the drain of the PMOS, and the VDD value for vtgm cell.

If the `-pmos_sweepvol` option is specified, ensure that a list of five values is set; otherwise, the tool returns an error and exits the process. If the `-pmos_sweepvol` option is not specified, the following default list is used: `{1.5 -1.5 -0.01 1.45 1.5}`

`-nmos_sweepvol { <sweep_voltage_begin> <sweep_voltage_end> <sweep_voltage_step> <voltage_of_the_drain> <VDD> }`

Specifies the sweep input voltage information (begin, end, and step), the voltage for the drain of the NMOS and the VDD value for vtgm cell.

This command must be set before the char_library command.

Example

```
set_vtgm_cell \
    -cells {INV1 BUF1}\
    -pmos_sweepvol { 1.5 -1.5 -0.01 1.45 1.5} \
    -nmos_sweepvol {-1.5  1.5  0.01 0.05 1.5} \
```

```
Library(…){
    model_vtgm : 1.525
    ....
    cell (INV1){
        cell_vtgm : 1.2
    }
    cell (BUF1){
        cell_vtgm: 1.525
    }
}
```

split_library

Enables the split library feature that extracts the cells from a library and writes them into the output directory in gzip compressed format. These libraries can then be used with commands such as interpolate_library to speed up the runs by parallelizing on a per cell or PVT basis.

Syntax

```
split_library \
     -directory <output_directory> \
     [ -cell <cell_name> ] \
     <library_file>
```

Options

`-directory <output_directory>`

Specifies the output directory.

`-cell <cell_name>`

Specifies a list of specific cell names.

`<library_file>`

Specifies the name of the library file.

unset_var

Resets the specified parameter to its default value. This command is equivalent to setting `set_var <parameter_name> [get_var_default <parameter_name>])`.

Options

<parameter_name>	Specifies the parameter name for the default value needs to be reset.

write_datasheet

Writes a datasheet in the format specified by the `-format` option.

Options

`-cells` Writes specified cells to the datasheet.

`-conditional` Includes writing out all conditional arcs. Default: write out data in the default groups only.

`-dir <dir_name>` HTML directory name.

`-exclude` Excludes specified cells from being written to the datasheet.

`-filename <file_name>`

Output file name.

`-format <text | html | pdf | ps>`

Datasheet format. Default: `"text"`

If the format is `text`, the datasheet is written to a file with the name specified with the `-filename` option. If no filename is specified, the datasheet is written to a file called `library_name.txt` by default.

The text-formatted datasheets support writing of truth tables too.

`-groups` Writes only specified cell groups to the datasheet.

`-include_indices` Include indexes in HTML reports.

`-logo <file_name>` Logo in GIF or JPG format.

`-map { list }` List of name-map pairs that are used to map an internal name to an external name. Default: none (which means there is no name mapping.)

`-table_style` Specifies where to create the datasheet tables from first-mid-last or min-avg-max. Default: first-mid-last

`-user_truth_table_dir <dir_name>`

Specifies a directory where truth tables are stored for inclusion in the datasheet. If this option is specified, Liberate will search for a user-provided truth table file with a name such as `cellName.txt`. The file format is not fixed. If the file is found, all the contents in that file will be copied into the datasheet.

Note: This command should be specified after the read_library command has been run.

<library_name> Library name for the datasheet and filename prefix.

This command can be used after char_library, read_ldb, and read_library.

The datasheet includes library information for each cell group. A cell group can be specified using the define_group command or it can be inferred from the footprint attribute. The cell group information includes the name of each cell in the group, the logic function, the pin capacitance, the area and any relevant leakage, delay, power and timing constraint information. The delay, power and constraint information is written as a table that includes the minimum, middle and maximum entry from the respective characterized table values.

If the html format is requested, the -dir directory name must also be specified. A collection of *<cell_group>.html* files will be written to this directory. The datasheet can be viewed by opening the *<dir>/index.html* file with an internet browser. If the -logo option is specified, a reference to the logo (gif or jpg) will be included at the top of each groups html file. The logo file must exist in *<dir>/<logo>*. If a schematic symbol for a cell group is available in the file *<dir>/sym/<cell_group>.jpg* then it will also be included. The datasheet writer is written using Tcl API of Liberate and is available for customization in the file $ALTOSHOME/etc/datasheet.tcl.

Note: if the cell group include only one cell, the symbol picture should be named as *<cell_group>.jpg*. However, if the cell group includes more than one cell, the symbol picture should be named as *<cell_group>x.jpg*.

The -format option also supports the values pdf and ps for writing out PDF and PostScript (.ps) format files. This is done by converting the HTML datasheet output first to postscript and then to PDF using two tools: html2ps and ps2pdf. These tools need to be installed and in the command *PATH* for postscript and PDF generation to work. They are available for download from the internet. The recommended setting is html.

The -include_indices option can be specified to request Liberate to include the indexes for delay, power and constraint tables when the format html is generated.

By default, with the -table_style option, the datasheet tables are created from the first point in any table, the last point, and the mid point. If set to min-avg-max, the min, avg, and max values from the table are used.

The -map option provides for mapping internal signal names to an external name. Options are provided as a list of signal name pairs. Example:

```
write_datasheet -map { intA myA intB myB }
```

Signal "intA" will be output as "myA" in the datasheet. ("intB" and any other signals will be handled similarly.)

The write_datasheet command outputs the group description defined by the define_group command in the HTML datasheet format. If the -conditional option is

used, all conditional arcs are also written out. The `write_datasheet` command will also create the directory specified by `-dir`, unless it already exists.

This command must be used after a database has been loaded (see char_library, read_ldb, and read_library).

Sample output

```
Delay(ns) to Q rising (conditional):
Cell NameTiming Arc(Dir)WhenDelay(ns):MinMidMax
DFF3TCLK->Q (RR)-0.02610.04650.1452
    SET->Q (FR)CLK&D0.04390.06600.2019
    SET->Q (FR)CLK&!D0.04580.06590.2026
    SET->Q (FR)!CLK&D0.03240.05330.1671
```

Examples

```
# Output the new datasheet in text format to tt.txt
write_datasheet tt

# Write a HTML datasheet
write_datasheet -format html -dir tt_html \
    -logo liberate.gif tt
```

write_ldb

Creates a library database (LDB). The LDB can then be used in a later Liberate session for formatting the library data, for example, creating a datasheet or generating a Verilog file.

Note: When this command is used in the multi-PVT characterization flow of Liberate Trio, the LDB structure will be changed by an additional level of hierarchy for each PVT that is characterized. For detailed information, see:

- Chapter 9, "Using the Multi-PVT Characterization Flow of Liberate Trio." of this reference manual, and
- Liberate Trio Licensing section in *LIBERATE Software Licensing and Configuration Guide*

Options

`-overwrite`	Disables the automatic version control, and if the output library already exists, it is overwritten. **Note:** This option is supported only in single PVT flows. For multi-PVT characterization flow of Liberate Trio, the `write_ldb -overwrite` option is disabled.
`-rename`	Renames an existing LDB file. The default is to not rename the previous LDB and create a unique file name for the new LDB instead.
<filename>	A library database file in ldb format.

Liberate will automatically save each cell as it is characterized to an LDB in the current directory named *altos.ldb.<#>*, where # is the process ID. The `write_ldb` command renames this file to the name given in the `write_ldb` command.

Important

It is recommended that the `write_ldb` command should be executed immediately after the char_library command and before any model creation commands such as write_library. This is highly recommended to ensure that a clean, unmodified copy of the LDB is saved for future use. This is important because, for example, when user data is loaded with `write_library -user_data`, the user data will modify the internal database and any LDB subsequently saved will contain those modifications.

If the gzip utility from GNU exists in the search path, the LDB will be gzipped automatically. This library database is named `<filename>.gz`.

In the single PVT characterization flow, when the `write_ldb allcells.ldb.gz` command is run, the following behavior can be observed:

- First run creates a `allcells.ldb.gz` file.
- Second run creates a `allcells.ldb.1.gz` file.

Similarly, when the `write_ldb -overwrite allcells.ldb.gz` command is run, the following behavior can be observed:

- First run creates a `allcells.ldb.gz` file.
- Second run removes the previous `allcells.ldb.gz` file and creates a new `allcells.ldb.gz` file.

In the multi-PVT characterization flow, as the `-overwrite` option is not supported, a warning message is displayed and the following behavior can be observed:

- First run creates the `allcells.ldb.1.gz` file.
- Second run retains the previously existing `allcells.ldb.1.gz` file and creates a new `allcells.ldb.2.gz` file.

Examples

Example 1

```
# characterize the library
char_library
# save the library database to tt.ldb
write_ldb tt.ldb
```

Example 2

```
# After finishing multi-PVT characterization, save the database to a network path
char_library -extsim spectre
write_ldb $path_to_mpvt
```

write_library

Outputs the library in Liberty format using Tcl API routines. (See *ALAPI Reference Manual*). The Tcl script for formatting the library is provided in the file `$ALTOSHOME/etc/format_library.tcl`. The `write_library` command should be specified in this file.

Note: When writing a library with noise data (ECSMN/CCSN), it is recommended to always include advanced timing (ECSM/CCS) data, that is, always include `-ecsm` (`-ccs`) when using `-ecsmn` (`-ccsn`). This is because the static timers desire a complete set of advanced model data for accurate timing analysis.

For compressing the Liberty files, you can use the Ascava Distillation tool. It produces compression of Liberty files up to twice of what the `tar` or `gzip` utility produces, and in a shorter run time. For usage information, see Appendix E, "Using Ascava Distillation Tool for Ultra Compaction of Liberate Files."

Options

`-bus_syntax {<>} | {[]} | {()}`

Specifies the bus syntax characters used when outputting the library and the `bus_naming_style` attribute. (See also the `-expand_buses` option.)

`-capacitance_only`

Disables the output of `rise_capacitance` and `fall_capacitance` attributes. The output library will only have a single capacitance attribute.

Note: This option is useful for backward compatibility. Care should be taken when using this option.

`-capacitance_range < 0 | 1 | 2 >`

Requests the output of `rise/fall_capacitance_range` attributes into the library. The supported values are:

- `0`: Omit.
- `1`: Include the rise and fall range spanning from the min of the `min_capacitance` values to the max of the `max_capacitance` values. This method has been reported to cause timing issues in PrimeTime. (Default)
- `2`: Include the rise and fall capacitance ranges where both range limits are both set to the rise/fall capacitance attribute values:

  ```
  rise_capacitance_range = "<rise_capacitance>,
  <rise_capacitance>"
  fall_capacitance_range = "<fall_capacitance>,
  <fall_capacitance>"
  ```

`-ccs` Include CCS data.

`-ccs_compact` Outputs a library in the compact CCS format.

`-ccs_compact_lc` Generates a compact CCS output by feeding the Liberate output library into Library Compiler.

`-ccs_precision` Controls the precision used when writing CCS waveform data into the generated library. The value for this option must conform to the standard Tcl formatting. Default: "%g"

`-ccs_voltage_waveform`

Includes CCS driver voltage waveforms in the output library. The construct is known as `normalized_output_voltage_waveform`. The `-ccs` option must also be used. Default: Do not include.

`-ccsn` Include CCSN (noise) data.

`-ccsp` Include CCSP (power) data.

`-ccsp_compact` Outputs a library in the compact CCSP format.

`-cells { cell_names }`

Specifies which cells get written to the output library. If `-exclude` is also set, the cells listed in the `-cells` list will be excluded from the output library file.
Default: all cells get written

Note: This option supports the use of a wildcard.

`-cell_cleanup <gzip | delete>`

Runs a cell-level library cleanup utility after successful generation of full libraries in MPVT or arc packet flow (write_library_mode=1). If this option is not specified, by default, all cell-level libraries are saved and preserved. The option supports use of the following values:

- `gzip`: Compresses cell libraries under the `LIBS` directory. This method uses less drive storage, but requires longer access time for future `write_library` operations.
- `delete`: Removes cell-level libraries after full libraries have been generated successfully. Uses the least disk space, but requires the longest access time for future `write_library` operations.

`-dcnoise_abstol <tolerance>`

Specifies tolerance used to while merging similar DC noise templates. Default: `1e-6` (volts)

If the noise values are less than this tolerance, the templates will be merged into a single group.

`-dcnoise_prefix <prefix>`

Controls the prefix used for writing DC noise templates. Default: `"DC_"`

`-derive_max_capacitance`

Computes the pin-based `max_capacitance` simple attribute from the characterized values in the `rise_transition/fall_transition` tables. In order for this option to work, the `rise_transition/fall_transition` tables must be characterized with `index_1` values covering the `max_transition` which is specified by the control parameter `max_transition` or the `define_max_transition` command. If the `index_1` (transition) values do not cover the `max_transition`, then Liberate will issue a warning during the `write_library` phase. In addition, this option will honor all of the following option/commands as long as they are used in the same Tcl script:

```
define_max_transition
define_max_capacitance_attr_limit
define_cell -ignore_input_for_auto_cap
```

`-driver_waveform` Outputs normalized driver waveforms into the library. For the output to include the driver waveform, the LDB/VDB must contain the driver waveform data. If the Tcl contains multiple `write_library` commands, the first command using this option will enable the waveform output for all subsequent `write_library` commands. Normalized driver waveforms will not be output for user defined PWLs which are incompletely specified or use wildcards.

`-driver_waveform_size <value>`

Sets the number of voltage points in the normalized driver waveform `index_2`. The normalized waveform currently uses an arbitrary number of voltage points uniformly distributed from gnd to vdd. The number of points can be controlled using this option. Default: `500`

When the predriver_waveform parameter is set,

- the default `-driver_waveform_size` is the same as the value set with predriver_waveform_npts.
- and `-driver_waveform_size` is ***not defined*** with the `write_library` command, the voltage points is ***not distributed*** uniformly.
- and `-driver_waveform_size` ***is defined*** with the `write_library` command, even if it is the same as `predriver_waveform_npts`, the voltage points ***is distributed*** uniformly.

When the driver_waveform is defined in the template, that is, in the re-characterization flow using references,

- the default value of `-driver_waveform_size` is `500`, regardless of the number of points in the reference or template file.
- the voltage points will always be distributed uniformly in the generated normalized_driver_waveform.

`-ecsm` Include ECSM data.

`-ecsm_ccs` Requests a library that contains both CCS and ECSM timing constructs. This option can be combined with `"-ccs -ccsn -ccsp"` or `"-ecsm -ecsmn -ecsmp"`.

The CCS and ECSM noise and power formats cannot be merged in the output library. The correct usage is `"-ecsm_ccs -ccsn -ccsp"` (*recommended*) or `"-ecsm_ccs -ecsmn -ecsmp"`. Merging these formats produces a very large library. Due to the large size of a combined library, it is preferable to build a library with only CCS data for PrimeTime and only ECSM data for Tempus. Use the `-ecsm_ccs` option when a single library is desired for use with both timers.

`-ecsmn` Include ECSM noise data.

`-ecsmp` Include ECSM power data.

`-em` Include Electromigration data.

`-exclude` Exclude cells given by `-cells`.

`-expand_buses` Specifies the library should be output with individual pins and no buses.

`-filename <filename>`

Specifies the name for Liberate to write out the library. If `filename` is not specified the library is written to `<library_name>.lib`.

`-fix_dc` Enables correction of DC tables. See also ccsn_check_dc_tables.

`-gzip` Compresses the output file using `gzip`. If the output library file already exists, a warning will be given and a unique file name will be generated using the given name suffixed with a unique number. (See `overwrite` option.)

`-indent <number>` Specifies the number of spaces to indent by. Default: `2`

`-lvf` Enables LVF generation from the Liberate run.

Note: This command option is supported in unified characterization flow of Liberate Trio. For detailed information, see:

- Chapter 9, "Using the Multi-PVT Characterization Flow of Liberate Trio." of this reference manual, and
- Liberate Trio Licensing section in *LIBERATE Software Licensing and Configuration Guide*

`-map { list }` List of name-map pairs, used to map internal names to an external name. Default: none (No name mapping.)

The option modifies the final name used for the internal node in the library. It is usually the case that the internal pin node name contains character that can not be used in a `.lib` format. Therefore, it is required to use a simpler name for such internal pins.

`-overwrite` Disables the automatic version control, and if the output library already exists, it is overwritten.

`-precision <precision>`

Controls the precision used when writing out the library. The value for this option must conform to standard Tcl formatting. Default: `"%g"`

`-preserve_user_data_precision { }`

Tells Liberate to preserve the precision of attributes in the `user_data` file and not to apply the `precision` to them.

Note: If user data contains quotation marks, such as `myData("1.001")`, this will be preserved in the output.

`-remove_failed <none | data | cell>`

Instructs Liberate on how to handle failed characterization data when writing to the library. Default: "" (`data` for Liberate MX, Liberate AMS, and Liberate Variety; `none` for Liberate and Liberate LV.)

The following are the supported values:

- `none`: Does not remove any failed data. See the parameters mark_failed_data, mark_failed_data_replacement, and constraint_failed_value for available user controls.
- `data`: Removes the arcs with failed data from the cell in the output library.
- `cell`: Removes the cell that includes any arc with failed data from the output library.

`-rename` Instructs Liberate to test for the existence of the output library. If the output library exists, rename the existing file before writing the library. The existing file will be renamed using the next available unused numerical index. By default, the `write_library` command tests for the existence of the output library and if it exists, a warning is printed and the output is written to the next available unused numerical index.

`-scan_ccsn_remove` Remove CCSN from scan dummy cell.

`-scan_dummy_scale_power`

Enables power data scaling for power tables while removing scan pins. The define_cell command must include scan-related information such as the scan pins for this option to work.

`-scan_output_dummy` Converts sequential cells (latches, flops) to scan dummy cells by removing the scan pins. This option removes all scan pins from a cell and writes out the reduced cell.

`-sdf_cond_equals {"==" | "===" | "== logical" | ""}`

Specifies how `sdf_cond` attributes are written. Default: `""` (none)

The following table explains supported values:

Value	Output
`"=="`	`"a == 1'b1 && b == 1'b0 …"`
`"==="`	`"a === 1'b1 && b === 1'b0 …"`
`"== logical"`	`"a == 1 && b == 0 …"`
`default`	`"a  && ~b …"`

`-sdf_edges` Enables the output of the `sdf_edges` attribute.

`-sensitivity_file <`*`filename`*`>`

When this option is specified, Liberate reads the specified sensitivity file. The sensitivity file is created by Liberate Variety using the `write_variation` command with the `-format "sensitivity"` option. When this option is used, the OCV delay groups in the sensitivity file will be merged into the nominal timing library. Both the `add_margin -sensitivity_file` and `write_library -sensitivity_file` options can be used in the same run.

`-si` Include SI data.

`-skip { leakage | power | hidden_power | conditional_hidden_power | nochange }`

Disables the output of the specified data type into the output `.lib` file. Default: none (do not skip any data)

The characterization of power (see `char_library -skip {power}`) cannot be skipped when CCS data is desired since Liberate needs the hidden power simulations to generate the receiver pin caps. Specifying `hidden_power` skips the output of hidden power arcs. Specifying `conditional_hidden_power` skips the output of conditional hidden power arcs. This capability is useful when characterizing a library using different SPICE models for timing and power.

`-swap_index_order` Swaps the index order for 2D tables specified with the `-swap_index_order_templates` option. Default: Use the order specified by the `read_library`, `define_template` or `read_ldb` command.

`-swap_index_order_templates { list }`

Specifies LUT template types whose indexes need to be swapped. Default: `all`

It is a list of 2D templates processed by the `-swap_index_order` option. Valid values are one or more specific template names used in the library or all. Default is all 2D templates.

An example that swaps constraint indexes for all constraints that use a template named `constraint_template_3x3` is given below:

```
write_library -swap_index_order
-swap_index_order_templates { constraint_template_3x3 }
-filename test.lib test
```

`-sync_ldb` Forces Liberate to read the ldb on disk prior to writing. This option is used to address possible precision issues between the `char_library` and `read_ldb/write_library` flows.

`-thread <number>` Number of different CPU threads to use to write out the library. This option defines the maximum number of threads to use on the current machine. By default, Liberate uses single thread when formatting and writing out library files. Running on two or more threads will reduce the wall time needed to output library files. The number of threads should be chosen such that the machine will not be overloaded.

`-unique_pin_data` Outputs unique data, such as timing and power, for each bus or bundle member. It specifies that original pin names are to be used inside the *when* condition string without going through the post-processing of changing pin names to bundle names.

`-user_data <filename>`

Specifies user-provided data in Liberty format to be merged with the current library. This is useful for including non-characterized data such as wire-load models in the output library. Once this user-data is merged into the current library, all subsequent `write_library` commands will output the merged constructs as part of the output library. If this is not desired, then separate runs of Liberate consisting of read_ldb and `write_library` must be executed. Any valid construct that is present in the user-provided but not present in the current library database will be copied to the output library, with the following exceptions:

- Attribute `slew_derate_from_library` is not copied.
- Attributes `function`, `state_function` and `area` will override values in the current library.
- Groups `state_table`, `ff` and `latch` will override the equivalent groups in the current library.

`-vector <"stimulus">`

Vector stimulus used to simulate the leakage, each bit can be one of X | 1 | 0.

`-vector <vectors>` Outputs the user-defined sensitization vectors into the output library. The vectors specified using the `-vector` and `-prevector` options of the define_arc command are known as user-defined vectors.

<library_name> The output library name.

The `write_library` supports buses. Buses can be defined by `define_cell`, in the ldb, or by the `define_bus` command. For each defined bus, a bus template is created in the library header (group name "type"). A `bus_naming_style` attribute is also created. For each bus, all the timing, power, ccsn data, etc. for that bus pin is represented once under the bus group with only capacitance, min/max_transition attributes given for each pin.

Note: This can result in a loss in accuracy, as all the data is taken from the first bus bit (the from index).

The `-ccs`, `-ccsn`, `-ccsp`, `-ecsm`, `-ecsmn`, `-ecsmp`, `-em`, and `-si` options enable inclusion of Liberty CCS timing, CCS noise, CCS power, ECSM, ECSM noise, ECSM power, electromigration (EM), and SI data in the output library if it exists in the characterized database (ldb). By default, only leakage values and NLDM timing and NLPM power table data are written. ECSM and CCS data are not written into the same library.

Important

Due to differences in the power data liberty format for CCSP and ECSMP, `write_library` can output *either* power format in a Liberate run, *but not both*. If both power formats are required, then separate Liberate runs must be used with a `read_ldb/write_library` flow.

Creation of libraries containing `CCSP` and `ECSMP` data is not permissible in the same Liberate characterization run because both have a different power format. Therefore, when you run the `char_library` and `write_library` commands in a Liberate session, ensure that you have not specified both `-ccsp` and `-ecsmp` options. You can however use the `-ccs`, `-ccsn`, and `-ccsp` options in the same Liberate run. In addition, if the `-ccsp` option is set with the `char_library` and `write_library` commands, the voltage_map parameter is overwritten to `1`.

Examples

Example 1

```
read_ldb complete.ldb

# Output a library without SI and merge my.lib
write_library -ccs -ecsm -user_data my.lib no_si

# Output a library with SI but no ECSM or CCS
write_library -si -user_data my.lib si_only

# Output a library for cells "AND2" and "OR2"
# Output cells AND2 and OR2 only, no SI, CCS or ECSM
write_library -cells { AND2 OR2 } and2_or2_only

# Omit cells AND2 and OR2, no SI, CCS or ECSM
write_library -cells{ AND2 OR2 } -exclude no_and2_or2

# changed the sdf_cond style
write_library -sdf_cond_equals "===" sdf_equals.lib
```

Example 2

By default, that is, without `-unique_pin_data`:

```
        bus (DOUT) {
```

```
  bus_type : bus_DDR3_DOUT_2_0;
  pin (DOUT[2]) {
    direction : output;
  }
  pin (DOUT[1]) {
    direction : output;
  }
  pin (DOUT[0]) {
    direction : output;
  }
  timing () {
    related_pin : "CONTROL";
    timing_sense : non_unate;
    timing_type : rising_edge;
    cell_rise (delay_template_7x7) {
       .....
    }
    rise_transition (delay_template_7x7) {
       .....
    }
    cell_fall (delay_template_7x7) {
       .....
    }
    fall_transition (delay_template_7x7) {
       .....
    }
  }
  internal_power () {
    related_pin : "CONTROL";
    rise_power (power_template_7x7) {
       .....
    }
    fall_power (power_template_7x7) {
       .....
    }
  }
}
```

Example 3

With `-unique_pin_data`:

```
bus (DOUT) {
  bus_type : bus_DDR3_DOUT_2_0;
  pin (DOUT[2]) {
    direction : output;
    timing () {
      related_pin : "CONTROL";
      timing_sense : non_unate;
      timing_type : rising_edge;
      cell_rise (delay_template_7x7) {
          .....
      }
      rise_transition (delay_template_7x7) {
          .....
      }
      cell_fall (delay_template_7x7) {
          .....
      }
      fall_transition (delay_template_7x7) {
          .....
      }
    }
    internal_power () {
      rise_power (power_template_7x7) {
          .....
      }
      fall_power (power_template_7x7) {
          .....
      }
    }
  }
  pin (DOUT[1]) {
    direction : output;
    timing () {
      related_pin : "CONTROL";
      timing_sense : non_unate;
      timing_type : rising_edge;
      cell_rise (delay_template_7x7) {
          .....
```

```
    }
    rise_transition (delay_template_7x7) {
        .....
    }
    cell_fall (delay_template_7x7) {
        .....
    }
    fall_transition (delay_template_7x7) {
        .....
    }
  }
  internal_power () {
    rise_power (power_template_7x7) {
        .....
    }
    fall_power (power_template_7x7) {
        .....
    }
  }
}
pin (DOUT[0]) {
  direction : output;
  timing () {
    related_pin : "CONTROL";
    timing_sense : non_unate;
    timing_type : rising_edge;
    cell_rise (delay_template_7x7) {
        .....
    }
    rise_transition (delay_template_7x7) {
        .....
    }
    cell_fall (delay_template_7x7) {
        .....
    }
    fall_transition (delay_template_7x7) {
        .....
    }
  }
  internal_power () {
    rise_power (power_template_7x7) {
```

```
            .....
        }
        fall_power (power_template_7x7) {
            .....
        }
      }
    }
  }
```

write_template

Creates a Liberate 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 file name if the name does not ends with this file extension). The Tcl file includes all the necessary define_template and define_cell commands needed to run Liberate. This command provides a convenient way to use an existing library's templates to create the Tcl file to characterize a new library.

Note: When this command is run, it does not print identical define_arc commands for arc-based CCSN.

Options

`-abs_tol <value>` Tolerance for comparing templates. Default: 0.0

`-auto_index` Generates templates suitable for use with `-auto_index` option of char_library.

The indexes in the template will contain scale factors that are derived from the input library if the `-index_*` (that is, `-index_delay`, `-index_const`, `-index_mpw`, `-index_si`) options are not set. The first index ratio is `0` and represents the minimum index value. The last index ratio is `1` and represents the maximum index value. The intermediate index values are the ratios calculated using the following formula:

```
ratio = original_index_value/(max_index_value -
     min_index_value)
```

See the `char_library -auto_index` command option for more details.

If any of the `-index_*` options are specified in the write_template command, the template indexes use the ratios determined by Liberate. The table size of the indexes follow the specified values. If a value is not specified, a default table size is used as following:

- Delay or Power: `7x7`
- Constraint: `3x3`
- MPW: `3x2`
- SI: the number of si_immunity indexes for `WIDTHS×LOADS`

Default: the value specified using the `-index_delay` option. Specify one or more of the `-index_*` options to one of these defaults for backward compatibility to LIBERATE 16.1 ISR3.

`-ccsn` Set this to request define_arc commands for CCSN arcs in the output template.

`-cells { cell_names }`

Controls which cells should be written to the template. Default: *all cells*

This option supports the use of a wildcard.

Note: If `-exclude` is also set, the cells listed in the `-cells` list will be excluded from the output template file.

`-combine_rise_fall_index`

Create `index_1` and `index_2` ranges spanning the maximum/minimum values from both rise and fall arcs.

`-define_index` Generates a define_index command for each arc of a cell whose indexes differ from the default indexes for the cell. By default, write_template assumes all the data of the same type (delay, power etc.) within a cell uses the same template. For certain types of cells there may be distinct indexes for different paths within that cell, where the slew and loading conditions used for characterization are different.

Note: The `-define_index` option cannot be used with the `-auto_index` option.

`-define_max_trans` Generates define_max_transition commands in the template for any cell or pin whose maximum transition is different from the default maximum transition.

`-dir <directory_name>`

Specifies a directory that stores the templates split into sub-templates for cells or cell groups. Default: do not split

The files will be named in the following format:

`<dir>/<cell>_template.tcl`

Note: To create unique templates per cell group, specify the `-dir`, `-group_cells`, and `-unique` options.

`-driver_waveforms` Converts normalized driver waveforms into define_input_waveform in template.

`-exclude` Excludes the cells given by the `-cells` option.

`-expand_buses` Generate `define_cell` commands using `bus_syntax` if buses exist in the input library.

`-group <number>` Specifies the Tcl file will only contain 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 trail characterization run with a representative subset of the cells in the library. Default: `all`

`-group_cells` Generate unique template per cell group; use with the `-dir` option.

Combines cells belonging to the same group into a single file as <group_name>_template.tcl. (See group_attribute and define_group)

`-index_const "value"`

Generates the number of constraint indexes for "DATA×REFERENCE". Default: `3×3`

For related details, see also Using the index_* options.

`-index_delay "value"`

Generates the number of delay/power indexes for "SLEWS×LOADS". Default: `7×7`

`-index_mpw` Generate this number of mpw indexes for "SLEWS×LOADS". Default: `3×2`

`-index_si` Generates the number of si_immunity indexes for "WIDTHS×LOADS". Default: the value specified using the `-index_delay` option

`-io` Create templates with `define_arc` commands for I/O's.

The `-io` option additionally generates `define_arc` (including vectors), `define_leakage`, and `define_index` commands for use with IO cell characterization. For pad pins, a default `define_pin_load` template will also be generated that includes a pull-up voltage equal to twice (Tcl variable `template_pullup_voltage_scale`) the voltage on that pin and a pullup and pulldown resistance of 4000 ohms (Tcl variable `template_resistance`). The `-verbose` option is equivalent to the `-io` option with the exception that the `pin_load` will not be output.

`-map { list }`

List of name-map pairs, used to map internal names to an external name. Default: `none` (No name mapping.)

The `-map` option modifies the name used for the internal node in the library when writing the template file. It is usually the case that the internal pin node name contains character that can not be used in a `.lib` format. Therefore, it is required to use a simpler name for such internal pins. This option allows you to specify the name for the internal pin that needs to be used in the various definitions contained in the template file (`define_cell`, `define_arc`, and so on).

`-merge <"skip_delay">`

Generates a "merged verbose" template with merged *when* conditions. The option accepts the value `"skip_delay"`, which merges only `constraint`, `mpw`, and `hidden` power arcs. This option must be used together with the `-verbose` option. For example:

```
write_template -merge skip_delay -verbose
```

Note: When using a "merged verbose" template, you must also set the define_arc_preserve_when_string parameter to `1`.

`-mpw`

Generates MPW templates if MPW tables exist in the loaded library.

`-no_internal_supply`

Disables the default behavior of recognizing `internal_ground` and `internal_power` attribute in the `pg_pin` library group and writing out `define_cell` with the `internal_supply` constructs. Default: `off`

`-precision <precision>`

Controls the precision used when writing out the template. The value for this option must conform to the standard Tcl formatting. Default: `"%g"`

Note: When you set this option to increase the number of significant digits in the template, you might also need to adjust the settings of the ldb_precision parameter and the `write_library -precision` command.

`-pvt_filename <filename>`

	Specifies the name of the file in which the power supply settings of commands such as `set_vdd`, `set_gnd`, `set_pin_vdd`, and `set_pin_gnd` should be output. The information is extracted based on the `pg_pin` data in the library.
`-sdf_cond`	Writes define arcs with `sdf_cond` conditions for timing arcs. **Note:** The `-sdf_cond` and `-when_as_vector` options of the `write_template` command cannot be used together. Using them together generates an error message.
`-si`	Generates `si_iv_curve` and `si_immunity` templates for each cell. If the current library does include any signal integrity data, a default `si_iv_curve` template with 35 points (Tcl variable `template_siv_points`) is generated along with `si_immunity` templates for each unique delay template. The noise width (`index_1`) for the `si_immunity` template is created by multiplying the slew index (`index_1`) of the delay template by a factor of 10 (Tcl variable `template_slew_to_width`). If the switching power indexes are swapped and `index_1` points to load, the tool recognizes the slew index used by the switching power template and pads the hidden power template based on that. If the input library does not have a two-dimensional noise immunity template, a two-dimensional template will be created from the delay indexes.
`-skip { leakage }`	Disables the output of `define_leakage` commands into the template file. This option should only be used when the *Inside View* algorithm is enabled. It cannot be used with the `char_library -io` option that disables the *Inside View* algorithm. If there are no `define_leakage` commands loaded and the *Inside View* algorithm is not enabled, the resulting library will ***not*** have any leakage states characterized. Currently, out of all supported values, you can skip only "`leakage`".
`-sort_pinlist`	Accepts a value of `in_bi_ou`. When this option is enabled, the `define_cell -pinlist` option in the template will have the pins sorted as follows: `"input bidi output"`.

`-truth_table`

Specifies that a template file will be written 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 setting will be used. The output template might then be edited to define index values. The following options will be ignored when using this option: `-cell`, `-exclude`, `-io`, `-unique`, `-unique_power`, `-group`, and `-define_index`. For more information about the truth table syntax, see Truth Table Format.

`-unique`

Specifies that each cell will have its own unique set of template definitions, otherwise cells will share templates where the templates are identical. Two templates are deemed identical if they have the same type, the same number of indexes 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` option because the resulting library will not match the original library.

`-unique_power`

Generate unique power templates. (Default: *use same template indexes for delay and power*)

If `-unique_power` is used, the power templates can have indexes that are different from the timing templates. If the arc conditions for delay and power are identical, then only `define_arc` commands are written for "delay" as these will force both power and delay characterization. If the conditions are different then a full set of delay arcs and a full set of power arcs are written into the template. By default, the power templates will use the same indexes as the delay templates.

Using this option can have a significant impact on run time because additional simulations to characterize the power will be required.

`-unique_power_supply_name`

If the -pvt_filename option is specified, this option filters set_pin_vdd and set_pin_gnd commands when the supply_name is VDD, GND, or VSS.

Note: By default, set_pin_vdd and set_pin_gnd commands will be output for all supplies. It is recommended not to use this option to ensure that all pin supplies are written out.

-use_lu_table_name — Reuse the lu_table names from the original library in the Liberate define_template commands.

-user_attributes { *user_defined_attributes* }

Requests the addition of the specified attributes in define_arc commands in the template file. When this option is specified, a verbose template will be created automatically. The define_arc commands for the arcs containing the attribute don't_sim will include the -value and -value_trans options so as to reuse the existing arc data.

Note: At this time, the only supported user attribute is "dont_sim".

Example:

```
write_template -user_attributes {dont_sim} template.tcl
```

-verbose — Creates verbose templates with define_arc commands.

-verbose_include_switch_internal

Writes out define_arc commands for switch cell arcs that include internal pins.

-when_as_vector — Converts the *when* conditions into a vector sequence rather than having a -when for each define_arc.

<*filename*> — File name for the Tcl template.

This command should be used after char_library, read_ldb, read_library, or read_truth_table.

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_library 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.

When generating define_index commands (see the verbose, io and define_index options) write_template will increase the number of indexes to the same size as the default template if it is smaller. For example, a pad pin might have a default 7x7 template, but for some arcs, it might use 4x4. As define_index requires 7x7, write_template will pad the 4x4 indexes by inserting mid-points starting between the 1st and 2nd index, 2nd and 3rd, and so on until there are enough indexes.

If buses exist in the input library, the pins will be output in bus syntax in the define_cell -input, -output, -bidi, -clock, -async, and -pinlist options. The -expand_buses option is used to make write_template generate the define_cell commands without using bus_syntax. For example, a bus DOUT with 3-bits will be represented as:

```
define_cell -output { DOUT [2-0] }
```

instead of

```
define_cell -output { DOUT [2] DOUT [1] DOUT [0]}
```

The bus syntax used for the template can be changed by setting the bus_syntax parameter. For example, if the library uses "[]" and the SPICE netlist uses "<>", then using set_var bus_syntax "<>" before write_template will cause the bus definition to appear in the template using the <> syntax. For example, -output { DOUT[2-0] } will be output as -output { DOUT<2-0> }

Using the index_* options

The -index_constraint, -index_delay, -index_mpw, and -index_si can be used with the auto_index option to specify the required data size. These can also be used to change the data table size from the original .lib to a modified table. Liberate does not support 3D data tables. When the golden .lib file has 3D data tables (possibly due to a dependent load) for constraints (3D setup/hold table) and load dependent mpw (2D mpw table), the write_template command ignores complete the 3d tables. The -index_* options can be used to adjust the data template size. If the user-specified dimension is different from the library then appropriate warnings will be printed and the data size will be adjusted. For example:

```
#User wants a one dimensional mpw table: 4x1
write_template -index_mpw 4x1 -mpw -define_index new_data/templates.tcl
```

Using the -driver_waveforms option

The `-driver_waveforms` option writes out a template including `define_input_waveform` commands that can be used to recharacterize cells using the same input waveforms as the existing library. The normalized driver waveform data ***must*** be present in the source library.

This template can only be used with version LIBERATE **12.1 ISR1** or later.

The `driver_waveforms` written by `write_template` can be re-used for a different voltage level by changing the `-vdd_val` parameter to define_input_waveform.

Following is an example showing the use of `-driver_waveforms` in a `read_library`, `write_template` flow:

```
read_library my.lib
write_template -driver_waveforms template.tcl
```

Its output will appear as follows:

```
set PreDriver10_dot_5_colon_rise_0_dot_004_pwl  { 0.0 0 6.5e-13 0.065 1.175e-
12 0.215835 1.675e-12 0.34251 2.25e-12 0.470272 2.85e-12 0.587447 3.85e-12
0.755671 5e-12 0.920547 5.775e-12 0.942784 6.65e-12 0.960506 7.65e-12 0.974145
8.825e-12 0.984284 1.175e-11 0.995448 1.6975e-11 1 }


set PreDriver10_dot_5_colon_rise_0_dot_072_pwl  { 0.0 0 1.17e-11 0.065 2.115e-
11 0.215835 3.015e-11 0.34251 4.05e-11 0.470272 5.13e-11 0.587447 6.93e-11
0.755671 9e-11 0.920547 1.0395e-10 0.942784 1.197e-10 0.960506 1.377e-10
0.974145 1.5885e-10 0.984284 2.115e-10 0.995448 3.0555e-10 1 }


set PreDriver10_dot_5_colon_fall_0_dot_004_pwl  { 0.0 0 6.5e-13 0.065 1.175e-
12 0.215835 1.675e-12 0.34251 2.25e-12 0.470272 2.85e-12 0.587447 3.85e-12
0.755671 5e-12 0.920547 5.775e-12 0.942784 6.65e-12 0.960506 7.65e-12 0.974145
8.825e-12 0.984284 1.175e-11 0.995448 1.6975e-11 1 }


set PreDriver10_dot_5_colon_fall_0_dot_072_pwl  { 0.0 0 1.17e-11 0.065 2.115e-
11 0.215835 3.015e-11 0.34251 4.05e-11 0.470272 5.13e-11 0.587447 6.93e-11
0.755671 9e-11 0.920547 1.0395e-10 0.942784 1.197e-10 0.960506 1.377e-10
0.974145 1.5885e-10 0.984284 2.115e-10 0.995448 3.0555e-10 1 }


define_template -type delay \
        -index_1 { 0.004 0.072 } \
        -index_2 { 0.0013 0.0142 } \
        delay_template_2x2

define_template -type power \
        -index_1 { 0.004 0.072 } \
        -index_2 { 0.0013 0.0142  } \
```

```
        power_template_2x2

define_cell \
      -input { A1 A2 A3 B1 B2 B3 } \
      -output { ZN } \
      -pinlist { A1 A2 A3 B1 B2 B3 ZN } \
      -delay delay_template_2x2 \
      -power power_template_2x2 \
      AOI33D4

define_input_waveform -slew_index 0.004 -dir fall -pwl
$PreDriver10_dot_5_colon_fall_0_dot_004_pwl -vdd_val 1.05 -gnd_val 0 -scale
-pinlist { AOI33D4 A1 AOI33D4 A2 AOI33D4 A3 AOI33D4 B1 AOI33D4 B2 AOI33D4 B3 }

define_input_waveform -slew_index 0.072 -dir fall -pwl
$PreDriver10_dot_5_colon_fall_0_dot_072_pwl -vdd_val 1.05 -gnd_val 0 -scale
-pinlist { AOI33D4 A1 AOI33D4 A2 AOI33D4 A3 AOI33D4 B1 AOI33D4 B2 AOI33D4 B3 }

define_input_waveform -slew_index 0.004 -dir rise -pwl
$PreDriver10_dot_5_colon_rise_0_dot_004_pwl -vdd_val 1.05 -gnd_val 0 -scale
-pinlist { AOI33D4 A1 AOI33D4 A2 AOI33D4 A3 AOI33D4 B1 AOI33D4 B2 AOI33D4 B3 }

define_input_waveform -slew_index 0.072 -dir rise -pwl
$PreDriver10_dot_5_colon_rise_0_dot_072_pwl -vdd_val 1.05 -gnd_val 0 -scale
-pinlist { AOI33D4 A1 AOI33D4 A2 AOI33D4 A3 AOI33D4 B1 AOI33D4 B2 AOI33D4 B3 }
```

Examples

```
read_library my.lib
# Output a Liberate Tcl command file with templates
set template_slew_to_width 15
write_template -si my_template

# Output a Liberate Tcl command file for auto_index
write_template -auto_index -index_delay 8x8 ai_template

# Output a Liberate Tcl command file IO cell char
write_template -io io_template
```

write_top_netlist

Creates a Verilog file that contains an instantiation of each cell within the library. The output is written to the specified output Verilog <*filename*>. If the given <*filename*> does not end with a `.v`, the suffix `.v` is added automatically. Each instance and pin is named as follows: `inst_`<*cell*>, `pin_`<*cell*>`_`<*pin*>

Options

`-cells { `*list_of_cells*` }`

List of cells to include in the output. Default: all cells

Note: If the `-exclude` option is specified, the list of cells specified by the `-cells` option will be excluded from the output file.

`-exclude` Exclude the list of cells. Default: treat cells list as include list

`-pin_prefix <`*string*`>`

Prefix for each pin. Default: `pin_`

`-instance_prefix <`*string*`>`

Prefix for each instance. Default: `inst_`

`-module <`*name*`>` Top module name. Default: `top`

<*filename*> Name of the output Verilog file.

The top-level Verilog can be used to verify SDF timing back-annotation as follows. Read the top-level Verilog generated by `write_top_netlist` into a timing tool along with the .lib from write_library. Generate an SDF file from the timer. Read the SDF file, the top-level Verilog plus the Verilog (from write_verilog) or Vital (from write_vital) into a gate-level simulator. The gate-level simulator will report any SDF annotation errors.

This command must be used after a database has been loaded.

write_training_data

Writes an encrypted training database file of normalized DC tables based on the library that the write_library command outputs. Therefore, the `write_training_data` command must be used after running the `write_library` command.

Options

`-filename <`*`filename`*`>`

Specifies a unique name for the training database file.

<*libname*> Specifies a unique name for the training database library.

In the subsequent characterization runs, you can use the read_training_data command to read the training database files generated using the `write_training_data` command and accordingly fix bad DC tables in a characterized library.

To ensure that your training database is updated with good tables found during the CCSN DC table checks, set the update_training_data parameter to `1` before running the `write_library` and `write_training_data` commands.

Examples

1. DC table fixes:

```
read_training_data $dbList
read_library $lib
write_library … -ccsn -fix_dc $lib
```

2. Generate a new training database / merge existing databases:

```
# Enable to read both training databases and pure .lib files.
read_training_data $dbList ;
# Liberate reads and checks the data, and then creates one new database.
write_training_data $newDb
```

3. Update the training database on the fly while checking the `write_library` data:

```
read_training_data $dbList
set_var update_training_data true
read_library $lib
write_library … -ccsn -fix_dc $lib
write_training_data $newDb
```

write_userdata_library

Takes a Liberty file as input, and writes out a Liberty-formatted file containing only the cells, attributes, and groups specified by the user. The purpose of this command is to create a userdata file for use with the write_library command.

Options

`-cells { list }`	List of cells to write
`-exclude { list }`	List of cells to exclude from -cells list
`-include_attributes { list }`	List of attributes to include.
`-exclude_attributes { list }`	List of attributes to exclude.
`-include_groups { list }`	List of groups to include.
`-exclude_groups { list }`	List of groups to exclude.
`<filename>`	User data filename for output

This command maintains a list of default groups and attributes to include, as well as a default list of groups and attributes to exclude. *(NOTE: the default lists to "include" and "exclude" do not contain the same items. See list below.)*

If you do not explicitly specify items to include or exclude, the output file contains only the defaults. To add or subtract from this list, you must specify the groups/attributes you wish to include (or exclude). The command also supports the keyword "default" as a convenient way of referring to the default lists. *(See examples.)*

The list of attributes to exclude does *not* apply to the attributes under groups that are included. If a group is included then *all* the attributes and sub-groups of that group are also included.

- Default include attributes:

```
area
cell_footprint
clear
clear_preset_var1
```

```
clear_preset_var2
clock
clocked_on
clock_gate_clock_pin
clock_gate_enable_pin
clock_gate_out_pin
clock_gate_test_pin
clock_gating_integrated_cell
data_in
direction
dont_touch
dont_use
enable
function
input_map
input_voltage_range
internal_node
is_level_shifter
level_shifter_data_pin
level_shifter_enable_pin
level_shifter_type
next_state
nextstate_type
output_voltage_range
power_down_function
preset
signal_type
state_function
table
three_state
```

- Default exclude attributes:

```
capacitance
cell_leakage_power
input_voltage
max_capacitance
max_transition
min_pulse_width_low
min_pulse_width_high
output_voltage
```

- Default include groups:

```
ff
latch
statetable
test_cell
```

- Default exclude groups:

```
hyperbolic_noise_above_high
hyperbolic_noise_below_low
hyperbolic_noise_high
hyperbolic_noise_low
input_voltage
output_voltage
pin
propagation_lut_template
```

Examples

```
# Include all the default groups plus the "input_voltage" group
read_library myLibrary.lib
write_userdata_library -include_groups { default input_voltage } userData.lib

# Exclude all the default groups plus the "statetable" group
read_library myLibrary.lib
write_userdata_library -exclude_groups { default statetable } userData.lib
```

write_vdb

Creates a vector library database (VDB) file for the current library.

Important

The `write_vdb` command is currently incompatible with a non-zero value for `packet_clients`. If your flow uses both of these, make sure that `packet_clients` is set to `0` immediately prior to the `write_vdb` command.

Options

`-auto_index`	Instructs Liberate to automatically generate table indexes for all constructs (except `si_immunity`) overriding the values specified in the given templates. The number of entries for each index is taken for the appropriate pre-defined template. This feature uses the `-max_transition` parameter to determine the range of output loads for each cell. To automatically generate `si_immunity` indexes set the max_noise_width parameter.
`-ccsn`	Include CCSN (noise) data.
`-cells { list_of_cells }`	List of cells to include in the output. Default: all cells
`-extsim <name>`	Name of external simulator to use. For related details, see Using the -extsim option.
`-io`	Enable IO mode.
`<vdb_filename>`	Name of the output VDB file.

The VDB file created by the `write_vdb` command includes vector data that is created during the preprocessing stage in Liberate. This file can be used to speed up preprocessing by storing the processed vector data and library structure in the VDB file.

Typically, the VDB that is created is used to drive separate characterization runs, each designed to process the same library with different corners (process, voltage, temperature.) The characterization script for a given run would first load the database with a **read_vdb** command. Once loaded, the char_library command will use the vector and structure information stored in the VDB file, and will not rerun the vector processing. This is true for both the server and the client processes.

For generating a VDB, the `write_vdb` command takes the place of `char_library` in a Tcl command file that has been fully set up to characterize a library.

Note: `char_library` should not be executed in the same run as `write_vdb`.

The `-ccsn` and `-io` options are used to generate and store the CCSN templates, structures, and vectors into the VDB in the normal and in the IO mode. This VDB can be read back in using `read_vdb` and used in the subsequent `char_library` flow for CCSN generation.

Note: CCSN in libraries across corners is based on internal node consistency, Using extractions and netlists with varying wire names in the `write_vdb` and `char_library` stages of the flow will produce unreliable results and is not supported.

This command must be used after a database has been loaded.

Using the -extsim option

The `-extsim` option instructs Liberate to use an external SPICE simulator instead of Alspice. Alspice is the default built-in SPICE simulator. The license for the external simulator must be available. Currently, Spectre is one of the supported external simulators.

Important

The setting for `-extsim` must correspond to the setting for the extsim_cmd parameter.

When the `-extsim` option is specified, temporary run directories named `altos.<unique_id>.0`, `altos.<unique_id>.1`, and so on will be created to store the external simulation run-time files based on the thread number (`0`, `1`, and so on). The `<unique_id>` is a unique ID based on the date, time, and the Liberate process ID.

If distributed processing is requested using the set_client command, these temporary run directories will be created in the directory specified by the `-dir` option of the `set_client` command.

If distributed processing is requested using the packet_log_filename parameter, the `TMPDIR` environment variable and the `tmpdir` parameter determine where the temporary simulation files will be stored. If the specified directory does not exist, Liberate tries to create it using the `mkdir -p` system command.

write_verilog

Creates a Verilog file for the current library. The file is saved with the name given with the <*verilog_filename*> option. If this name does not have a `.v` file extension at the end, a `.v` suffix will be added automatically to the filenames.

Options

`-add_default_udp` Allows you to provide user data where all the primitives are not included; default primitives will be used for all other cells. This option works only with `-user_data`.

`-cells { cell_names }`

Controls which cells should get written to the output. Default: *all cells*

This option supports the use of a wildcard.

Note: If `-exclude` is also set, the cells listed in the `-cells` list will be excluded from the output Verilog file.

`-compare` Enables function comparison using the `-conformal` option of the `compare_function` command of Liberate LV. To use this flag, you must have license to run Liberate_LV and Conformal (Liberate_LV_Server, Liberate_LV_Client, and LEC licenses). In addition, the environment must be setup to enable these tools.

`-delayed` Controls naming convention for creating *delayed* signals. Default: `"delayed_%P"` (where `%P` is the pin name.)

`-exclude` Excludes the cells specified by using the `-cells` option.

`-fwire_prefix <prefix>`

Specifies the prefix used for internal wires created when generating logic functions. Default: `int_fwire_`

`-indent <number>` Specifies the number of spaces to use for indentation. Default: `tab`

`-merge` Merges cell modules from `user_data`.

The merge operation includes the cells that are not specified in the library, but are present in the `user_data` file.

`-mpw_include_output_state`

Includes the logic state of the output in the condition for MPW checks on "clear" and "preset" signals in the generated Verilog. Default: *do not use*

This option requires the sdf_cond_style parameter is set to `1`. If this parameter is not set, the `-mpw_include_output_state` option will force this setting.

Note: To ensure consistency between the library and the Verilog file, this parameter should be set prior to creating an equivalent library (`.lib`) with write_library. Otherwise, there might be warnings during SDF back-annotation.

The `-mpw_include_output_state` option should be used *before* read_library, char_library, or read_ldb. When this option is used, the MPW check (`$width`) will be checked by Verilog only when the output pin of the cell is high for clear inputs and low for preset inputs. The Verilog will contain extra "timing" gates to add the logic necessary to "*and*" the logic state of the output pin to logic representing the "*when*" condition given in the library for each `min_pulse_width` timing arc.

`-mux`

Instructs to use MUX primitives instead of basic logic primitives to represent MUX functions. Default: *basic logic primitives*

This option converts pins whose functions are a 2x1 or 4x1 MUX into a predefined, user-defined primitive (UDP) named *altos_mux2* and *altos_mux4* respectively.

`-notifier`

Sets the name of the Verilog register that is used to flag timing violations. Default: `"notifier"`

Note: This option is needed only if `-user_data` is used.

`-no_edge`

Excludes 'posedge' or 'negedge' on edge-triggered arcs. Default: *include edges*

This option will change the following:

```
specify
    if ((CP & SN))
        (negedge RN => (QN-:1'b0)) = 0;
```

...to this:

```
specify
    if ((CP & SN))
        (RN => QN) = 0;
```

`-no_err_primitives`

Causes `*_err` primitive cells to be replaced with `buf` cells.

This is option is available because some tools such as ATPG do not accept `*_err` primitives.

`-path <path>`

Controls the delimiter used for delay assignments. It must be either `"=>"` or `"*>"`. Default: `"=>"`

`-power_pin`

Adds the power-down function to the Verilog model. This option is used when the power-down function is needed in `liberty.lib` and the power pins should be included in the cell-port list.

`-sdf_version <version>`

Controls the format of the output Verilog for use with SDF annotation. SDF version must be `2.1` or `3.0`. Default: `2.1`

Setting this option to `3.0` generates a Verilog file that is compatible with SDF version 3.0. SDF version 3.0 permits *recrem* constructs in the Verilog to represent recovery and removal of timing constraints.

`-specparams`

Causes delay assignments in the Verilog to be assigned to *specparam* variables rather than directly to values.

`-split_nonunate` Instructs Liberate to convert a non_unate timing group in the .lib into two arcs in the Verilog output file.

This option is useful when using the Cadence ETS static timer to generate the SDF for annotation into the Verilog simulation. ETS will convert the single non_unate timing group into 2 SDF entries. This can cause an error condition in Verilog if the Verilog has only one non_unate arc and the SDF has two-delay entries.

`-split_notifier` When writing Verilog modules for multi-bit cells, it is required to output separate notifier commands for each `DFF`. The `-split_notifier` option is used to output separate notifier commands for each `DFF`.

`-table_style <min-avg-max | first-mid-last>`

Instructs Liberate to add real timing data into the Verilog file using the specified specparams syntax. Default: `""` (do not use real timing).

This option specifies the syntax to be used in the specparams. When this is not used, Liberate defaults to writing the Verilog with zero (0) for all timing and will expect that a Standard Delay Format (SDF) file is supplied from the STA tool.

`-timescale "`*`timescale_value`*`"`

Change the timescale when writing out Verilog. Default: `"1ns/10ps"`

`-twire_prefix <`*`prefix`*`>`

Specifies the prefix used for internal wires created when generating additional functions for state-dependent timing constraints. Default: `int_twire_`

`-udp_prefix <`*`prefix`*`>`

Specifies the prefix used for built-in user-defined primitives (UDPs) that are created for latches and/or flip-flops. Set `-udp_prefix` to a null string (`""`) to exclude generating UDPs. Default: `altos_`

`-use_liberate_function`

Generates Liberate functions when user data does not contain any function. If user data has functions and this option is specified, Liberate generates the functions again.

Note: This option works with the `-user_data` option.

`-user_data <filename>`

Specifies a user-provided Verilog file to merge with the generated Verilog data. If a `user_data` file is provided, timing information (paths and any additional wires required to specify the conditions for those paths) are merged with the user file and written to the output file, replacing any existing user-provided timing information. Without a `user_data` file, a complete Verilog file is written including function descriptions.

<verilog_filename>

Name of the output Verilog file.

The `write_verilog` command should *not* be used in the same run as the char_library, read_ldb, or write_library command. Instead, it should be used in a separate Liberate run after a read_library command. This is because the Liberty file might have formatting that is required for the Verilog output to be properly formatted. For example:

```
read_library my.lib
write_verilog my.v
```

Important

This command must be used after a database has been loaded.

Using the -delayed option

The `-delayed` option controls the naming convention for creating "delayed" signals. When using user-data with `write_verilog`, it is necessary to match these delayed signals with the equivalent signals used in the user-provided functional description. By default, delayed output signals are created for the signals passed to timing checks such as setup-hold and/or recrem in Verilog.

The delayed option uses a special variable `"%P"` to return the pin name and combine it with a user-defined string. For example:

```
write_verilog -delayed "delayed_%P"
```

If there is no `"%P"` and the `-delayed` string begins with `"_"`, it is used as a suffix. In the following example, the wire for the pin `"CK"` is `"CK_mySuffix"`.

```
write_verilog -delayed "_mySuffix" ...
```

if there is no `"%P"` and the `-delayed` string does not begin with `"_"`, the string is treated as a prefix. In the following example, the wire for the pin `"CK"` is `"myPreFixCK"`.

```
write_verilog -delayed "myPreFix" ...
```

For a pin named `"myPin"`, this will produce a delayed signal name `"delayed_myPin"`. Some examples are below:

Example 1:

```
module DFFSRN (QN, D, CP, RN, SN);
    output QN;
    input D, CP, RN, SN;
    reg notifier;
    wire delayed_D, delayed_CP, delayed_RN, delayed_SN;

    // Function
    ....
    // Timing
    specify
        ...
        $setuphold (posedge CP, posedge D, 0, 0, notifier,,, delayed_CP, delayed_D);
        ...
        $recrem (posedge RN, posedge CP, 0, 0, notifier,,, delayed_RN, delayed_CP);
        ...
    endspecify
endmodule
```

Example 2:

```
write_verilog -delayed "dly_%P"
```

... will produce:

```
    wire dly_D, dly_CP, dly_RN, dly_SN;
    ...
    $setuphold (posedge CP, posedge D, 0, 0, notifier,,, dly_CP, dly_D);
```

Example 3:

```
write_verilog -delayed "%P_d"
```

... will produce:

```
    wire D_d, CP_d, RN_d, SN_d;
```

```
        ...
        $setuphold (posedge CP, posedge D, 0, 0, notifier,,, CP_d, D_d);
```

The default name for these delayed signals is "delayed_<*pin_name*>" (for example, delayed_%P) where <*pin_name*> is a pin that is involved in a timing check.

To disable generation of delayed signals, use "" (empty double quotes), as illustrated below:

```
module DFFSRN (QN, D, CP, RN, SN);
    output QN;
    input D, CP, RN, SN;
    reg notifier;

    // Function
    ....
    // Timing
    specify
        ...
        $setuphold (posedge CP, posedge D, 0, 0, notifier);
        ...
        $recrem (posedge RN, posedge CP, 0, 0, notifier);
        ...
    endspecify
endmodule
```

Tcl Variables to Control the Verilog Output Format

The following Tcl variables can also be used to control the format of the Verilog output:

verilog_delay_value	The delay value. Default: 0
verilog_delay_Zvalue	The delay value for tristates. Default: 0
verilog_delay_clk2q_value	The delay value for clock to Q arcs on sequential cells. Default: 0
verilog_IQ	The name map for the internal state of flip-flops (for example, IQ) to the Verilog state function.
verilog_IQN	The name map for the internal state of flip-flops (for example, IQN) to the Verilog state function.

`verilog_start_skip`	Line to mark the start of the timing section in the `user_data` file which is to be replaced. The specified value must match exactly apart from the leading or trailing white space. Default: `"specify"`
`verilog_stop_skip`	Line to mark the end of the timing section in the `user_data` file which is to be replaced. The specified value must match exactly apart from the leading or trailing white space. Default: `"endspecify"`

Example

```
read_library my.lib
# Output a Verilog file
write_verilog -user_data my_verilog my.v
```

write_vital

Creates a Vital VHDL file for the current library. The file is saved with the name given with the <*vital_filename*> option. If this name does not have a `.vhd` file extension at the end, a `.vhd` suffix will be added automatically to the filenames. The output will include sequential gates and tristates.

Options

`-cells { cell_names }`

Controls which cells should get written to the output. Default: *all cells*

This option supports the use of a wildcard.

Note: If `-exclude` is also set, the cells listed in the `-cells` list will be excluded from the output Vital VHDL file.

`-compare`

Enables function comparison using the `-conformal` option of the `compare_function` command of Liberate LV. To use this flag, you must have license to run Liberate_LV and Conformal (Liberate_LV_Server, Liberate_LV_Client, and LEC licenses). In addition, the environment must be setup to enable these tools.

`-component <filename>`

Instructs Liberate to print a list of components into the specified output file. By default, no component file is output.

`-exclude`

Excludes the cells given by using the `-cells` option.

`-indent <number>`

Specifies the number of spaces to use for indentation. Default: `tab`

`-merge`

Merges cell modules from `user_data`.

The merge operation includes the cells that are not specified in the library, but are present in the `user_data` file.

`-no_edge`

Excludes 'posedge' or 'negedge' on edge-triggered arcs. Default: *include edges*

This option will change the following:

```
specify
    if ((CP & SN))
        (negedge RN => (QN-:1'b0)) = 0;
```

...to this:

```
specify
    if ((CP & SN))
        (RN => QN) = 0;
```

`-sdf_version <2.1 | 3.0>`

Specifies the desired SDF version in the output file. SDF version must be `2.1` or `3.0`. Default: `2.1`

Setting this option to `3.0` generates a Vital file that is compatible with SDF version 3.0.

`-user_data <`*`filename`*`>`

Specifies a user-provided Vital file to merge with the generated Vital data. If a `user_data` file is provided the port names and functional behavior information is extracted from the user-specified file. Without a `user_data` file, a complete Vital file is written. The sections in the Vital file that are extracted from the user data are marked with comments indicating that they were user-provided and not automatically generated, for example:

```
--  FUNCTIONALITY SECTION (USER PROVIDED) --
--  END FUNCTIONALITY SECTION (USER PROVIDED) --
```

`<vital_filename>` Name of the output Vital (vhd) file.

The `write_vital` command should ***not*** be used in the same run as the char_library, read_ldb, or write_library command. Instead, it should be used in a separate Liberate run after a read_library command. This is because the Liberty file might have information that is required for the Vital output to be properly formatted. For example:

```
read_library my.lib
write_vital my.vhd
```

Important

This command must be used after a database has been loaded.

Tcl Variables to Control the Vital Output Format

The following Tcl variables can be used to control the format of the Vital output:

`vital_config`	Include config section. Default: `0`
`vital_port_type`	Port type. Default: `STD_LOGIC`
`vital_delay_value`	The delay value. Default: `0ns`
`vital_delay_Zvalue`	The delay value for tristates. Default: `0ns`

`vital_delay_variables`

Flag to enable assignment to `Delay` variables instead of values. Default: `1`

`vital_timing_violation_format`

The prefix for `Violation` variables used by setup/hold timing checks. Default: `"Tviol_\$count"`, where `$count` is incremented for each `VitalSetupHoldCheck` entry within a cell

`vital_recrem_violation_format`

The prefix for `Violation` variables used by recovery/removal timing checks. Default: `"Rviol_\$count"` where `$count` is incremented for each `VitalRecoveryRemovalCheck` entry within a cell

`vital_pulse_violation_format`

The prefix for `Violation` variables used by pulse/period timing checks. Default: `"Pviol_\$count"` where `$count` is incremented for each `VitalPeriodPulseCheck` entry within a cell

`vital_timing_info_format`

The prefix for `TimingData` variables used by setup/hold timing checks. Default: `"SetupHoldInfo_\$count"` where `$count` is incremented for each `VitalSetupHoldCheck` entry within a cell

`vital_recrem_info_format`

The prefix for `TimingData` variables used by recovery/removal timing checks. Default: `"RecoRemoInfo_\$count"` where `$count` is incremented for each `VitalRecoveryRemovalCheck` entry within a cell

`vital_pulse_info_format`

The prefix for `PeriodData` variables used by period/pulse timing checks. Default: `"PeriodDataInfo_\$count"` where `$count` is incremented for each `VitalPeriodPulseCheck` entry within a cell

`vital_removal_check` The prefix to use for `Entity` variables used in removal checks. For SDF `3.0` annotation change this variable to `"removal"`. Default: `"hold"`

`vital_start_architecture`

The keyword that indicates the start of the architecture section in the `user_data` file. Default: `"architecture"`

`vital_stop_architecture`

The line that indicates the end of the architecture section in the `user_data` file. Default: `"end \$cell\_arch"` where `\$cell` is substituted with the current cell name

`vital_start_cell` The line to indicate the start of a vital cell description in the `user_data` file. Default: `"-- %BEGIN \$cell"` where `$cell` is substituted with the current cell name

`vital_stop_cell` The line to indicate the stop of a Vital cell description in the `user_data` file. Default: `"-- %END \$cell"` where `$cell` is substituted with the current cell name

`vital_start_function`

The line to indicate the start of a Vital function description in the `user_data` file. Default: `"function"`

Note: The spaces and dashes (`-`) and case will be ignored.

`vital_stop_function` The line to indicate the stop of a vital function description in the user data file. Default: `"\$vital_path_delay"` where `$vital_path_delay` is substituted with the value of the `vital_path_delay` variable, (Default: `"VitalPathDelay01Z"`)

Tcl Variables to Customize Formatting of Generated Timing Check Variables

The following Tcl variables can be used to customize the formatting of the generated timing check variables:

`$count` The current count of `VitalSetupHoldCheck`, `VitalRecoveryRemovalCheck`, or `VitalPeriodPulseCheck` entries.

`$ip` The input pin name.

`$rp` The reference pin name.

`$edge` The edge of the input pin transition.

`$ref_edge` The edge of the reference pin transition.

Note: The above-listed Tcl variables must be preceded with a "\" and suffixed with "\\" when used in the middle of a string to avoid early evaluation by the Tcl interpreter.

Example

```
read_library my.lib
# set timing check variables
set vital_timing_violation_format "Tviol_\$ip\\_\$rp\\_\$ref_edge"
set vital_recrem_violation_format "Rviol_\$ip\\_\$rp\\_\$ref_edge"
set vital_timing_info_format      "Tinfo_\$ip\\_\$rp\\_\$ref_edge"
set vital_recrem_info_format      "Rinfo_\$ip\\_\$rp\\_\$ref_edge"
# Output a Vital file
write_vital -user_data my_vital my.vhd
```

5

Liberate Parameters

This chapter describes the Liberate-specific parameters that impact library creation.

List of Liberate Parameters

The Liberate-specific parameters are set using the set_var command.

To access officially supported context-sensitive help information on a command or a parameter from within the tool, follow the procedure covered in the Invoking Liberate Help section.

To review tool-wise support information about each command and parameter available in the Liberate characterization portfolio, see *Liberate Characterization Portfolio Command and Parameter Support Matrix*.

a...	
■ active_operating_pvt	■ alspice_option
■ add_arc_index_to_ldb	■ auto_index_distinct_risefall
■ add_margin_info	■ auto_index_input_slew
■ adjust_tristate_load	■ auto_index_load_ratio_cap
■ adjust_tristate_load_ccsp	■ auto_index_weak_driver_mode
■ alspice_diode	■ auto_test_cell
■ alspice_leakage_option	

ccs_...	
■ ccs_abs_tol	■ ccs_infer_output_dir
■ ccs_base_curve_points	■ ccs_init_voltage_comp_thresh
■ ccs_base_curve_share_mode	■ ccs_max_current_thresh
■ ccs_cap_duplicate_risefall	■ ccs_max_pts
■ ccs_cap_enhancement_format_mode	■ ccs_multiple_switching_output_mode
■ ccs_cap_hidden_pin	■ ccs_process_post_ldb
■ ccs_cap_hidden_pin_mode	■ ccs_rel_tol
■ ccs_cap_is_propagating	■ ccs_segmentation_effort
■ ccs_cap_is_propagating_select_one_mode	■ ccs_simplify_thresh_mode
■ ccs_cap_mode_add_missing	■ ccs_smooth_lower_rise
■ ccs_cap_non_hidden_pin	■ ccs_smooth_upper_fall
■ ccs_cap_use_input_transition	■ ccs_voltage_smooth_thresh
■ ccs_cap_use_input_transition_tristate	■ ccs_voltage_tail_tol
■ ccs_correct_current_by_area	■ ccs_voltage_tail_tol_mode
■ ccs_current_model_pin_load	■ ccs_voltage_tail_trim_tol
■ ccs_enable_sawtooth_out	■ ccs_voltage_waveform_style
■ ccs_force_grid_delay	■ ccs_warn_negative_rcvr_caps
■ ccs_from_ecsm_linear_interp_factor	■ ccs_waveform_min_time_step
■ ccs_from_ecsm_smooth_mode	■ ccs_waveform_smooth_mode
ccsn_...	
■ ccsn_allow_duplicate_condition	■ ccsn_miller_init_vin_thresh
■ ccsn_allow_multiple_input_switching	■ ccsn_miller_slew_width
■ ccsn_allow_overlap_when	■ ccsn_miller_vin_mode
■ ccsn_allow_partial_voltage_swing	■ ccsn_miller_vout_delta_variation
■ ccsn_arc_channel_check	■ ccsn_model_channel_connected_ccbs
■ ccsn_arc_consistent_cut	■ ccsn_model_passgate_compatibility_mode
■ ccsn_arc_high_effort	

- ccsn_bus_holder_mode
- ccsn_channel_inputs_high_effort
- ccsn_check_data
- ccsn_check_data_max_dc_current_min_thresh
- ccsn_check_dc_tables
- ccsn_check_non_peak_noise_prop_range
- ccsn_consistent_side_inputs
- ccsn_controlling_path_check
- ccsn_dc_static_check
- ccsn_dc_static_check_mode
- ccsn_dc_static_check_thresh
- ccsn_dc_template_size
- ccsn_default_conditional_check
- ccsn_default_group_add_when
- ccsn_default_group_criteria_mode
- ccsn_dual_tie_enable
- ccsn_extra_default_stages
- ccsn_filter_probe_mode
- ccsn_first_stage_load1
- ccsn_first_stage_load2
- ccsn_floating_init_mode
- ccsn_include_passgate_attr
- ccsn_io_allow_multiples
- ccsn_io_mode
- ccsn_io_mode_enable
- ccsn_load_cap_delta_variation
- ccsn_load_cap_mode
- ccsn_model_related_node_attr
- ccsn_no_input_dc_current_mode
- ccsn_one_sided_tristate
- ccsn_part_mode
- ccsn_pin_criteria_mode
- ccsn_pin_high_effort
- ccsn_pin_stage_lshift
- ccsn_pin_stage_merge_mode
- ccsn_pin_voltage_level_attrib
- ccsn_prefer_min_vt_probe
- ccsn_prefer_pin_termination
- ccsn_print_is_needed_if_false_attr_value
- ccsn_probe_enable_toggle_res_check
- ccsn_probe_non_gate
- ccsn_prop_noise_check_criteria_mode
- ccsn_prop_noise_peak_mode
- ccsn_prop_retry_duration_incr
- ccsn_prop_retry_peak_incr
- ccsn_prop_tristate_meas_mode
- ccsn_prune_last_stage
- ccsn_related_power_pin_check
- ccsn_simultaneous_switch_save_vecdata
- ccsn_slew_based_output_voltage
- ccsn_sort_merge_hidden_mode
- ccsn_switch_cell_partition_mode
- ccsn_uda_user_probe_mode
- ccsn_use_io_ccb_format

■ ccsn_load_cap_slew_width ■ ccsn_load_cap_tran_tend ■ ccsn_merge_equivalent_stage_tol ■ ccsn_miller_init_mode	■ ccsn_use_output_voltage_level ■ ccsn_vector_define_mode ■ ccsn_vout_output_precision ■ ccsn_xfr_ccc_probe_mode
ccsp_...	
■ ccsp_base_curve_points ■ ccsp_default_group ■ ccsp_intrinsic_res_criteria ■ ccsp_leakage_current_abstol ■ ccsp_leakage_current_compensation_mode ■ ccsp_meas_supply_cap_sim_duration ■ ccsp_meas_supply_cap_ramp_ratio ■ ccsp_min_pts ■ ccsp_pin_direction_post_default ■ ccsp_prune_factor	■ ccsp_prune_second_tol ■ ccsp_prune_start_tol ■ ccsp_quantization_num_steps ■ ccsp_rel_tol ■ ccsp_related_pin_mode ■ ccsp_segmentation_effort ■ ccsp_table_reduction ■ ccsp_tail_tol ■ ccsp_unateness_infer_mode
ce...	
■ cell_port_case	
ch...	
■ char_mos_term_cap	■ char_mos_term_cap_skip_names
cl...	
■ cleanup_tmpdir	■ client_pending_timeouts
com...	
■ combinational_out_to_out_arc	■ combinational_risefall

cpu...	
■ cpu_load_threshold	■ cpu_memory_rel
■ cpu_memory_min	
d...	
■ datasheet_truthtable_in_pin_limit	■ discard_timing_sense_after_merge
■ def_arc_drive_side_bidi	■ disk_wait_time
■ def_arc_msg_level	■ driver_cell_acc_mode
■ def_arc_vector_consistency_check	■ driver_cell_all_inputs
■ default_power_avg_mode	■ driver_cell_info
■ default_rcvr_cap_groupwise	■ driver_cell_load_all_outputs
■ define_arc_ignore_mode	■ driver_cell_load_ldb_cmd
■ define_arc_preserve_when_string	■ driver_cell_trim_miller
■ define_duplicate_cap_mode	■ driver_waveform_arcs_only
■ define_input_waveform_check_action	■ driver_waveform_output_precision
■ delay_constrained_by_setup_recovery	■ driver_waveform_pulse_mode
■ delay_inp_fall	■ driver_waveform_slew_index_tolerance
■ delay_inp_rise	■ driver_waveform_wildcard_mode
■ delay_min_max_mode	■ duplicate_pin_attr_mode
■ delay_out_fall	■ duplicate_risefall_power
■ delay_out_rise	■ duplicate_risefall_power_ccsp
■ disable_method	
e...	
■ ecsm_arctype_enable	■ em_vector_gen_mode
■ ecsm_cap_hidden_pin	■ em_window_estimate_mode
■ ecsm_cap_input_slew_mode	■ enable_advance_licensed_features
■ ecsm_cap_load_effect_tol	■ enable_command_history
■ ecsm_cap_mode	■ enable_network_and_health_checks

m...	
■ mac_address_query_timeout	■ mega_mode_constraint
■ mark_failed_data	■ mega_mode_delay
■ mark_failed_data_replacement	■ mega_mode_hidden
■ max_capacitance_attr_limit	■ mega_reduced_function_mode
■ max_capacitance_attr_mode	■ mega_reverse_mos_mode
■ max_capacitance_auto_mode	■ mega_short_circuit_mode
■ max_capacitance_derive_limit_maxload	■ merge_related_preset_clear
■ max_capacitance_factor	■ min_capacitance_for_outputs
■ max_capacitance_limit	■ min_output_cap
■ max_hidden_vector	■ min_period
■ max_leakage_vector	■ min_period_when
■ max_noise_width	■ min_transition
■ max_transition	■ min_transition_attr_limit
■ max_transition_attr_limit	■ min_transition_factor
■ max_transition_factor	■ min_transition_for_outputs
■ max_transition_for_outputs	■ min_transition_include_power
■ max_transition_include_power	■ minimum_memory_warning_limit
■ max_transition_include_rcvr_cap	■ model_vth
■ measure_cap_active_driver_mode	■ mpw_criteria
■ measure_cap_lower_fall	■ mpw_glitch_peak
■ measure_cap_lower_rise	■ mpw_input_threshold
■ measure_cap_upper_fall	■ mpw_linear_waveform
■ measure_cap_upper_rise	■ mpw_search_bound
■ measure_ccs_cap_lower_rise	■ mpw_search_mode

r...	
■ ramp_vsrc	■ res_tol
■ ramp_vsrc_mode	■ reset_leakage_current_mode
■ ramp_vsrc_ratio	■ reset_negative_constraint
■ rc_floating_cap_mode	■ reset_negative_delay
■ rcp_cmd	■ reset_negative_leakage_power
■ rdb_checkpoint_dir	■ reset_negative_leakage_power_value
■ rdb_delete_upon_completion	■ reset_negative_path_delta_measurement
■ rdb_exit_if_source_differ	■ reset_negative_power
■ rechar_chksum	■ resolve_collision
■ removal_glitch_peak	■ retry_count
■ res_merge	■ retry_count_file_operation
■ res_open_tol	■ rsh_cmd
s...	
■ scalar_power_warning	■ ski_sync_method
■ scale_load_by_template	■ ski_use_large_memory
■ scale_tran_by_template	■ skip_nfs_sync
■ scan_dummy_include_leakage_power	■ slew_lower_fall
■ sdf_cond_equals	■ slew_lower_rise
■ sdf_cond_postfix	■ slew_normalize
■ sdf_cond_prefix	■ slew_upper_fall
■ sdf_cond_style	■ slew_upper_rise
■ sdf_cond_style_for_constraints	■ sort_cells
■ sdf_cond_variable_map	■ sort_groups_under_pin
■ sdf_logic_and	■ sort_pins
■ sdf_logic_not	■ sort_pins_under_when
■ sdf_logic_or	■ spectre_dash_log
■ server_timeout	■ spectre_use_char_opt_license

- set_var_failure_action
- sim_default_engine
- sim_duration
- sim_estimate_duration
- sim_init_condition
- sim_init_condition_estimation_mode
- sim_init_duration
- sim_power_duration_extend
- sim_use_init_duration
- simultaneous_switch
- simultaneous_switch_offset
- simultaneous_switch_use_arc_when
- simultaneous_switch_worst_vector
- ski_alter_mode
- ski_clean_mode
- ski_compatibility_mode
- ski_enable
- ski_mdlthreshold_exact
- ski_meas_mode
- ski_power_subtract_output_load_match_extsim
- ski_reset_cnt
- spectre_use_mmsim_token_license
- spice_character_map / spectre_character_map
- spice_delimiter
- spice_delimiter_replacement
- spice_instance_name_require_x_prefix
- spice_logical_netname_mode
- subtract_hidden_power
- subtract_hidden_power_consider_all_supplies
- subtract_hidden_power_scalar_mode
- subtract_hidden_power_use_default
- supply_define_mode
- supply_info
- switch_cell_bounded_dc_current
- switch_cell_dc_current
- switch_cell_dc_current_output_offset
- switch_cell_internal_net_name
- switch_cell_internal_node_timing_arc
- switch_cell_powerdown_function
- switch_function_attr_mode

t...

- template_unique_power_mode
- test_cell_at_end
- test_cell_filter_attributes
- timing_group_unateness
- tmpdir
- toggle_leakage_state
- tran_tend_estimation_mode
- tristate_disable_transition
- tristate_pin_cap_always_on_res_mode

active_operating_pvt

Holds the value of the currently active PVT as set by the set_pvt command. It can be used in the scripts to get the currently active PVT. It must not be used to set the active PVT; use the `set_pvt` command instead.

Example

```
define_pvt ...
set_pvt ...
set active_pvt [get_var active_operating_pvt]
```

The `get_var` command will return a valid PVT only after the `set_pvt` command has been executed.

add_arc_index_to_ldb

`<0 | 1>` Controls whether the arc indexes (IDs) should be added to the LDB.
Default: `0`

`0`	Disables adding of arc IDs to the LDB.
`1`	Adds arc IDs to the LDB.

This command must be used before the read_ldb command is run.

When a cell is not characterized fully, that is, the RDB file is incomplete, the recovery flow automatically runs the arcs for which the RDB files are missing. However, the tool does not rerun characterization for the successful arcs, that is, the ones for which RDB files were generated. Recharacterization of these successful arcs is possible based on their arc IDs, which can be saved to the LDB if the `add_arc_index_to_ldb` parameter is set to `1`.

add_margin_info

`<0 | 1>`

Controls whether an `add_margin` report should be generated while applying the user-specified `add_margin` commands.
Default: `0`

- `0` — Applies the user-specified `add_margin` commands without generating an `add_margin` report.
- `1` — Applies the user-specified `add_margin` commands and generates an `add_margin` report.

  The report is saved by filename, `add_margin.log`. It contains the formulas and computations that explain how the data in the margined library was computed. The report can be useful to understand the application of user-specified additional margin.

This parameter must be set prior to generating a library with the `write_library` command.

adjust_tristate_load

`<0 | 1 | 2 | 21 | 22>`

Controls whether pin capacitance is added to the load indexes 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 indexes when outputting the library. (Default)

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 indexes for the index_2 values. When using the write_template command, the pin capacitance is subtracted from the load indexes 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.

By default, Liberate adds the pin capacitance of the tri-state pin to each of the load indexes when outputting the library. The rise index_2 adds the rise_capacitance and the fall index_2 adds the fall_capacitance. In addition, when using the write_template command to create a Liberate Tcl command file, the tri-state pin rise_capacitance and fall_capacitance is subtracted from the load indexes specified in the input library to create the appropriate define_template commands for tri-state pins.

This parameter must be used before the char_library command.

Note: This parameter can be used after char_library, but must be used before any models are generated.

Example:

```
# Disable adjusting tristate pin load indexes
set_var adjust_tristate_load 0
```

adjust_tristate_load_ccsp

`<0 | 1>` Controls adjusting the load index (`index_2`) of CCSP dynamic waveforms for tri-state load. This parameter must be used with the adjust_tristate_load parameter.
Default: `0`

- `0` Do not adjust the load indexes of the CCSP current waveforms.
- `1` Adjust the load indexes in the CCSP current waveforms by the tristate capacitance. Follow the algorithm set by the `adjust_tristate_load` parameter.

This parameter must be set before the write_library command is run.

allow_one_shot_jobs

`<0 | 1>` Enables or disables creation of a one-shot job for a small cell.
Default: `1`

Note: This parameter works when Bolt job distribution system is used in multi-PVT mode.

- `0` Disables the feature. No one-shot job is created and jobs are submitted in the following order for all small and large cells regardless of their arc size:
  1. counting
  2. precharacterization, if enabled
  3. simulation
  4. assembly

`1`	Enables the feature. Consequently, unlike a large cell that goes through both simulation and assembly jobs, a one-shot job is created for a small cell. This saves resources because a small cell, which might have only one arc to characterize, need not be submitted for both simulation and assembly jobs to finish characterization.

This parameter must be set before the char_library command is run.

alspice_diode

`<0 | 1>`	Controls the use of an improved algorithm for handling diodes in Alspice. Default and recommended: `1`
`0`	Enables you to achieve the behavior of releases prior to 3.1
`1`	Enables the algorithm for handling diodes in Alspice.

This parameter must be used before the `read_spice` command.

alspice_leakage_option

`{`*`options`*`}`	Functions the same as alspice_option, except that it is targeted for leakage simulations and override values set by alspice_option. The available options are: `sim_method`, `sim_gmin`, and `sim_step`. The recommended setting is `sim_method` to override the default homotopy.

This parameter must be used before the char_library command.

Example:

```
# Set the leakage options for Alspice
set_var alspice_leakage_option "sim_method=trap sim_gmin=1e-14 sim_step=1e-14"
```

alspice_option

`{options}`

Accepts a list of simulation control options to be used by Alspice for delay, power, and timing constraint characterization. The options are specified in the format `"name=value"`.
Default: There are no fixed default value. Alspice determines appropriate values on a case to case basis.

Note: This parameter has no effect on simulations performed by an external simulator.

`sim_method` Specifies the integration method. The valid values are: `"trap"` and `"gear"`.

Note: The recommend setting for this is either `trap` or `gear`. This is in order to override the default homotopy, which is a proprietay algorithm.

`sim_gmin` Specifies gmin for Alspice.

`sim_step` Specifies time step for Alspice.

If the values for `gmin` and `tstep` is not specified, Alspice examines the data and formulate its own determination about appropriate values. However, Alspice always uses the explicitly specified

This parameter must be used before the char_library command.

Example

```
# Set the simulation options for Alspice
set_var alspice_option "sim_method=trap sim_gmin=1e-12 sim_step=1e-12"
```

auto_index_distinct_risefall

`<0 | 1>`

Controls the computation of both the rising and falling maximum capacitance and load index values when the `-auto_index` option of the `char_library` command is used.
Default: `0`

This parameter must be set before the char_library command is run.

auto_index_input_slew

<value>	Specifies the value (in seconds) of the input slew to be used to calculate max_capacitance during auto_index characterization. The input slew is measured using the slew threshold values defined by the following parameters: slew_lower_rise, slew_lower_fall, slew_upper_rise, and slew_upper_fall. Default: max_transition

This parameter must be set before the char_library command is run.

auto_index_load_ratio_cap

<value>	Specifies a ratio value such that the computed geometric ratio is not permitted to exceed the specified value. That is, this parameter defines a limit for the geometric ratio used by the auto_index algorithm (see char_library -auto_index). Default: 10

Note: The specified ratio value should be as per the following guideline: 1.0001 < value <= FLT_MAX

If the desired geometric ratio exceeds the value set for this parameter, the excess might result in a large ratio between the first load index (see min_output_cap) and the second load index.

Set this parameter to 2.5 for backward compatibility with the LIBERATE 16.1 ISR4 and prior releases.

This parameter must be set before the char_library command is run.

auto_index_weak_driver_mode

`<0 | 1>` Controls if the `-auto_index` option of the `char_library` command adjusts `min_capacitance` of cells that cannot satisfy the `min_transition` at `0` load.
Default: `0` (Do not compensate).
Recommended: `1`

- `0` Prints a warning if `min_slew` does not meet the cell output load. Choose the last attempted valid load.
- `1` Same as 0. However, ensure that the methodology is consistent between Alspice and `extsim_exclusive` modes.

Some cells do not have sufficient drive strength to satisfy the `min_slew` condition when not loaded. Setting this parameter allows for better correlation between the `min_load` values generated during `-auto_index` between Alspice and `extsim_exclusive` characterizations.

This parameter must be used before the char_library command is run.

auto_test_cell

`<0 | 1>` Controls automatic creation of the `test_cell` attribute.
Default: `1`

- `0` Disables automatic creation of the `test_cell` attribute. Use this setting for backward compatibility with the LIBERATE18.1 ISR2 release.

`1` Enables automatic creation of the `test_cell` attribute.

A `test_cell` attribute can also be added in the `user_data` file specified with the write_library `-user_data` command or by using the set_attribute command.

The order of precedence is:

1. `set_attribute` command definition
2. `user_data` file
3. `auto_test_cell` creation

This parameter must be used before the write_library command is run.

binning_detail

`<low | medium | high | highest>`

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.

`highest` Results in the most detailed and largest library while still merging completely equivalent whens.

This parameter is used to set the criteria for determining how individual state-dependent groups are merged. This parameter affects the characterization and should be specified before the `char_library` command.

This parameter must be used before the char_library command is run.

Example

```
# Enable low binning detail
set_var binning_detail low
```

bisection_info

`<0 | 1 | 2 | 3 | 4>` Prints additional constraint bisection information.
Default and recommended: `0`

`0`	No information is provided. This is the default and recommended production setting because it minimizes file input/output and provides better runtime and disk usage.
`1`	External circuit simulation decks are annotated with bisection information. This mode forces `constraint_bisection_mode` to `0` (pure bisection). **Note:** Because many spice decks are utilized in a bisection search, it is recommended to use a setting of `2` or `3`, which provides additional information in the log file.
`2`	External circuit simulation decks are annotated with search information. The search method is not modified.
`3`	The search iteration information is added to the log file as a `LIB-405 message`. For debugging a search, this is the recommended setting.
`4`	Similar to mode 3, but adds the pin direction and path to decks for each iteration when the decks are written.

For constraint and MPW searches, circuit simulation decks can be annotated with comments that give the slew, alignment, and iteration number for each search iteration in that deck. In addition, the log file can contain, along with the measurement result for each search iteration, the same information as the deck. This parameter controls the information that is included.

Use this parameter for information about debugging constraint bisection searches. To obtain the circuit simulation decks, see the `extsim_deck_dir`, `extsim_save_passed`, `extsim_save_failed` parameters, and the `-extsim` argument of the `char_library` command.

Note: It is often easier to follow the debug information while characterizing a single slew or load.

This parameter must be used before the char_library command.

bolt_cell_priority_criteria

`< equal_bins | score_based | user_defined >`

Sets the order in which the cell simulation jobs are queued. Default: `equal_bins`

`equal_bins` Cells are sorted by rank and divided equally into six priority bins.

`score_based` Cells are sorted by rank, and then assigned to the bins where:

```
bin_size = (max_score - min_score)/6
```

`user_defined` User-defined CSV file specified by an absolute path in the `CDS_BOLT_USER_DEFINED_PRIORITIES` environment variable.

The format of the CSV file is `CELL,PRIORITY` on each line, with the `PRIORITY` ranging from 3 (lowest) to 8 (highest).

bolt_client_cpu_load_threshold

<value> Ensures that when bolt_client_health_checks is enabled (=1), any client is shut down if the average load on the client is above the threshold for bolt_client_cpu_load_time_threshold in seconds. It is not recommended to set this parameter below 1.2 because the full utilization rate on a healthy machine is in the 1.0 range. Default: 2.5

Note: This parameter is associated with the Bolt Job Distribution System. See also packet_arc_job_manager.

This parameter must be used before the char_library command is run.

bolt_client_cpu_memory_min

<value> Ensures that when bolt_client_health_checks is enabled (=1) and the available CPU memory on the client falls below the specified value, the client is killed.
Where *value*>0; Default: 1 (in GB)

Note: This parameter is associated with the Bolt Job Distribution System. See also packet_arc_job_manager.

This parameter must be used before the char_library command is run.

bolt_client_cpu_memory_rel

<value> Ensures that when bolt_client_health_checks is enabled (=1) and the percent of available memory falls below the specified value, the client is killed.
Where *value*>0; Default: 1 (% free memory)

Note: This parameter is associated with the Bolt Job Distribution System. See also packet_arc_job_manager.

This parameter must be used before the char_library command is run.

bolt_client_cpu_utilization_threshold

<value> Setting this parameter to a non-zero value enables an additional criteria to consider a CPU to be overloaded. If the average CPU load is found to be over the specified bolt_client_cpu_load_threshold value for a time longer than the `bolt_client_cpu_load_time_threshold` value, then the CPU utilization percentage is also checked to see if it is greater than the specified `bolt_client_cpu_utilization_threshold` value. For example, assume that this parameter is set to `80`. Then, if the utilization of the CPU is below `80%`, the CPU will not be considered overloaded even if the average CPU load is over the user-specified load threshold.

This feature is useful if the CPUs commonly have processes that run without utilizing the CPU fully, but still result in blocking the CPU resources (for example, running processes that are waiting for I/O operations), and therefore cause the average load to be high. If it is desired to run the Liberate client job in this scenario, that is, where the average load is high but the CPU utilization is low, then you can set the `bolt_client_cpu_utilization_threshold` parameter to a non-zero value and enable the additional criteria.

Valid values: An integer between `0` and `100` (percentage)

Default: `0` (means that the CPU utilization check is disabled)

Note: This parameter is associated with the Bolt Job Distribution System. See also packet_arc_job_manager. It works together with bolt_client_cpu_load_threshold and `bolt_client_cpu_load_time_threshold`.

This parameter must be used before the char_library command is run.

bolt_client_disk_space_min

`<value>` Ensures that when bolt_client_health_checks is enabled (`=1`) and the available disk space falls between the specified value and 10MB, a warning message is generated. If the disk space falls below 0.01GB, it is considered a hard fail and the client is killed.
Where *value*`>0.01`; Default: `1` (in GB)

Note: This parameter is associated with the Bolt Job Distribution System. See also packet_arc_job_manager.

This parameter must be used before the char_library command is run.

bolt_client_health_checks

`<0 | 1>` Enables the network and job health checks when using the Bolt Job Distribution System. See also packet_arc_job_manager.
Default: `0`

`0`	Disables the network and job health checks.
`1`	Enables all the network and job health checks.

This parameter must be used before the char_library command is run.

bolt_client_heartbeat_interval

`<value>` Specifies the time interval for the heartbeat message to be sent from the clients to the Bolt server to tell the server that the client is still alive.
Where *value*`>100` seconds; Default: `180` (Seconds)

Note: This parameter is associated with the Bolt Job Distribution System. See also packet_arc_job_manager.

bolt_client_pending_timeout

`<value>` Ensures that when bolt_client_health_checks is enabled (`=1`) and the client is still in the pending state after the specified time has elapsed, the client is killed.
Where *value*`>0`; Default: `50000` (Seconds)

Note: This parameter is associated with the Bolt Job Distribution System. See also packet_arc_job_manager.

This parameter must be used before the char_library command is run.

bolt_client_startup_status

`<true | false>` Specifies whether to enable tracking the status of the client startup.
Default: `0`

`true`	Enables the tracking.
`false`	Disables the tracking.

This parameter must be used before the char_library command is run.

In Liberate Trio mode, when the Bolt job distribution system is also enabled. you can track the status of the clients starting up on the remote hosts. This is done with a set of status files instead of communicating through the Bolt server because the clients are not yet online. If the clients remain in "pending" state for a long time (as defined by the bolt_client_startup_status_interval parameter), then a report of what is happening with each client is reported in the Liberate server log.

bolt_client_startup_status_interval

`<value>` Specifies how often the Liberate server should check the files to update the log. The time interval is set in seconds.
Default: `1800` (Seconds)

Note: This parameter is associated with the Bolt Job Distribution System. See also packet_arc_job_manager.

This parameter must be used before the char_library command is run.

A report is dumped to the log after the time specified using the `bolt_client_startup_status_interval` parameter has lapsed. The log output of the Liberate server looks like the one shown below when there are pending clients and the timeout interval has expired:

```
(Jun 23 17:32:27)
INFO (LIB-13004): There are 10 of 20 clients still in pending state:
INFO (LIB-13005): STATE                      COUNT  CLIENTS (max 5 shown)
INFO (LIB-13005): Checking-out license                  10
client_1,client_11,client_13,client_15,client_17
INFO (LIB-13005): Startup sequence completed             7
client_10,client_12,client_14,client_16,client_18
(Jun 23 17:33:27)
```

The Liberate tool does not take any action. The report is generated to notify the users to let them take an appropriate action, if any.

The following states are checked for and reported:

Pending in farm (or killed/died)	The client is still waiting in LSF/SGE in a PEND state for a host to run on; otherwise, the job was killed or it died before the Liberate launch wrapper script could be run.
Failed to start executable	The client was issued a host and it started running the Liberate launch wrapper script, but the Liberate executable could not start. The possible reasons could be one of the following: ■ The path to the executable is not available on the client host. ***Solution:*** Fix the host to get access to the NFS path where the Liberate executable is located. ■ The host on which the client is running is of an unsupported operating system (OS) type and therefore, the executable is failing to run. ***Solution:*** Fix your LSF/SGE bsub/qsub `rsh_cmd` parameter to specify the correct resource to acquire a host with a Liberate supported version of the OS. ■ Other configuration issues on the host causing the executable to fail to launch. ***Solution:*** Contact your IT department to debug the issues with this host.

Checking-out license	Liberate client is attempting to checkout a license. It might be waiting in a loop to acquire a license if all licenses are currently in use.
Connecting and sending heartbeat	Liberate checked out a license successfully. Liberate is now connecting to the Bolt server that is attempting to send a heartbeat to the Liberate server. If this continues for a long time, check the Bolt server setup and health. Make sure you have the correct Bolt settings and correct value specified with the `CDS_BOLT_SERVER` environment variable.
Startup sequence completed	The Liberate client started successfully and is ready to accept jobs from the Liberate server.

For all the scenarios given above, you can check the client log file for details about what is happening with that client's startup. For **pending in farm** however, a client log file might not be available because the job did not even start. In this case, check the LSF log file you specified with the `rsh_cmd` parameter.

bolt_connection_lost_random_interval

<integer>

Sets the time interval in seconds to wait before attempting to reconnect to the Bolt server. This parameter is useful to reduce multiple simultaneous connection requests, especially when a Bolt host goes down. The number of random interval seconds are a number between `0` and the value specified using this parameter.
Default: `10` (seconds)

The specified value must be `10` seconds or greater.

Note: This parameter is associated with the Bolt Job Distribution System. See also packet_arc_job_manager.

bolt_connection_retry_forever

`<0 | 1>`

Controls the reconnect attempts that should be made to the hosts in a Bolt server cluster.
Default: `0` (or `false`)

Note: This parameter is associated with the Bolt Job Distribution System. See also packet_arc_job_manager.

`0`	Retries to reconnect to all the hosts just once and quits if the reconnect fails. **Note:** Alternatively, you can specify the value as `false`.
`1`	Enables to keep trying forever to reconnect to the hosts in a Bolt server cluster. **Note:** Alternatively, you can specify the value as `true`.

bolt_connection_retry_interval

`<integer>`	Sets the time interval in seconds for each connection retry between the hosts within a single Bolt server cluster (or the same host if there is only one). Default: `10` (seconds) The specified value must be `10` seconds or greater. This parameter is associated with the Bolt Job Distribution System. See also packet_arc_job_manager.

bolt_connection_retry_timeout

`<integer>`	When set to a value larger than `0`, this parameter specifies the duration, in seconds, to retry a connection to the Bolt server. If the timeout is reached, Liberate Trio server/client will exit. Default: `0` (seconds), where `0` indicates disabled This parameter cannot be used with bolt_connection_retry_forever. **Note:** This parameter is associated with the Bolt Job Distribution System. See also packet_arc_job_manager.

Given below is an example of the output format of a Bolt connection retry when `CDS_BOLT_SERVER` is set to `linws41,linws42,linws43` and the following parameters:

```
set_var bolt_connection_lost_random_interval 20
set_var bolt_connection_timeout 30
set_var bolt_connection_retry_interval 10
```

```
set_var bolt_connection_retry_timeout 360
```

Notice that the number of retries per host is equal to `bolt_connection_timeout` divided by `bolt_connection_retry_interval`. In this case, applying this formula will translate to `30/10 = 3` (as you can see below the connection retries per host occur 3 times).

It is recommended that the `bolt_connection_retry_interval` and `bolt_connection_lost_random_interval` be set to a higher interval when using thousands or tens of thousands of clients per Bolt server as used by all Liberate users.

```
--
(Jan  2 19:51:27)    Waiting 13 seconds (randomized 20 seconds) before retrying
connection to next host...
(Jan  2 19:51:40)    Cannot connect to the communication server on host 'linws41',
retry count: 1 of 3...
(Jan  2 19:51:50)    Cannot connect to the communication server on host 'linws41',
retry count: 2 of 3...
(Jan  2 19:52:00)    Cannot connect to the communication server on host 'linws41',
retry count: 3 of 3...
(Jan  2 19:52:10)    Trying to connect to next host in list: linws43 ...
(Jan  2 19:52:10)    Waiting 13 seconds (randomized 20 seconds) before retrying
connection to next host...
(Jan  2 19:52:23)    Cannot connect to the communication server on host 'linws43',
retry count: 1 of 3...
(Jan  2 19:52:33)    Cannot connect to the communication server on host 'linws43',
retry count: 2 of 3...
(Jan  2 19:52:43)    Cannot connect to the communication server on host 'linws43',
retry count: 3 of 3...
(Jan  2 19:52:53)    Trying to connect to next host in list: linws42 ...
(Jan  2 19:52:53)    Waiting 13 seconds (randomized 20 seconds) before retrying
connection to next host...
(Jan  2 19:53:06)    Cannot connect to the communication server on host 'linws42',
retry count: 1 of 3...
(Jan  2 19:53:16)    Cannot connect to the communication server on host 'linws42',
retry count: 2 of 3...
(Jan  2 19:53:26)    Cannot connect to the communication server on host 'linws42',
retry count: 3 of 3...
(Jan  2 19:53:36)    -- All hosts in cluster tried 1 times for 0h 2m 10s (130 sec).
Retrying... (user defined retry timeout set to 360 sec)


(Jan  2 19:53:36)    Trying to connect to next host in list: linws41 ...
(Jan  2 19:53:36)    Waiting 13 seconds (randomized 20 seconds) before retrying
connection to next host...
(Jan  2 19:53:49)    Cannot connect to the communication server on host 'linws41',
retry count: 1 of 3...
(Jan  2 19:53:59)    Cannot connect to the communication server on host 'linws41',
retry count: 2 of 3...
```

```
(Jan  2 19:54:09)   Cannot connect to the communication server on host 'linws41',
retry count: 3 of 3...
(Jan  2 19:54:19)    Trying to connect to next host in list: linws43 ...
(Jan  2 19:54:19)    Waiting 13 seconds (randomized 20 seconds) before retrying
connection to next host...
(Jan  2 19:54:32)   Cannot connect to the communication server on host 'linws43',
retry count: 1 of 3...
(Jan  2 19:54:42)   Cannot connect to the communication server on host 'linws43',
retry count: 2 of 3...
(Jan  2 19:54:52)   Cannot connect to the communication server on host 'linws43',
retry count: 3 of 3...
(Jan  2 19:55:02)    Trying to connect to next host in list: linws42 ...
(Jan  2 19:55:02)    Waiting 13 seconds (randomized 20 seconds) before retrying
connection to next host...
(Jan  2 19:55:15)   Cannot connect to the communication server on host 'linws42',
retry count: 1 of 3...
(Jan  2 19:55:25)   Cannot connect to the communication server on host 'linws42',
retry count: 2 of 3...
(Jan  2 19:55:35)   Cannot connect to the communication server on host 'linws42',
retry count: 3 of 3...
(Jan  2 19:55:45)    -- All hosts in cluster tried 2 times for 0h 4m 19s (259 sec).
Retrying... (user defined retry timeout set to 360 sec)


(Jan  2 19:55:45)    Trying to connect to next host in list: linws41 ...
(Jan  2 19:55:45)    Waiting 13 seconds (randomized 20 seconds) before retrying
connection to next host...
(Jan  2 19:55:58)   Cannot connect to the communication server on host 'linws41',
retry count: 1 of 3...
(Jan  2 19:56:08)   Cannot connect to the communication server on host 'linws41',
retry count: 2 of 3...
(Jan  2 19:56:18)   Cannot connect to the communication server on host 'linws41',
retry count: 3 of 3...
(Jan  2 19:56:28)    Trying to connect to next host in list: linws43 ...
(Jan  2 19:56:28)    Waiting 13 seconds (randomized 20 seconds) before retrying
connection to next host...
(Jan  2 19:56:41)   Cannot connect to the communication server on host 'linws43',
retry count: 1 of 3...
(Jan  2 19:56:51)   Cannot connect to the communication server on host 'linws43',
retry count: 2 of 3...
(Jan  2 19:57:01)   Cannot connect to the communication server on host 'linws43',
retry count: 3 of 3...
(Jan  2 19:57:11)    Trying to connect to next host in list: linws42 ...
(Jan  2 19:57:11)    Waiting 13 seconds (randomized 20 seconds) before retrying
connection to next host...
(Jan  2 19:57:24)   Cannot connect to the communication server on host 'linws42',
retry count: 1 of 3...
(Jan  2 19:57:34)   Cannot connect to the communication server on host 'linws42',
retry count: 2 of 3...
```

```
(Jan  2 19:57:44)    Cannot connect to the communication server on host 'linws42',
retry count: 3 of 3...
(Jan  2 19:57:54)    Quitting retries since disconnected from server for more than
user specified timeout of 360 seconds
ERROR (LIB-464): (Jan  2 19:57:54 : unable to connect): The program will exit now
as an unexpected communication error occurred: host: linws41,linws43,linws42 :
T20200102195126p140424: Connection refused. Check that environment variable
'CDS_BOLT_SERVER' is set correctly and rerun the tool.
```

bolt_connection_timeout

`<integer>` — Sets the time interval in seconds after which a given server or client should exit if it is not able to connect to the Bolt server.
Default: `100` (seconds)

bolt_exit_after_preprocessing

`<0 | 1>` — Controls whether the tool should exit after finishing the preprocessing.
Default: `0`

Note: This parameter only impacts the Bolt-related flows. The legacy Packet Arc or Packet Cell flows are not impacted.

- `0` — Disables the feature.
- `1` — Liberate runs only the counting jobs and exits after the preprocessing is done. The access clients are killed as soon as they are not needed.

bolt_idle_client_timeout

`<integer>` — Defines the timeout when a client will be killed if it has not received a job for the given number of seconds.
Default: `86400` (that is, 1 day)

bolt_kill_clients_in_foreground

`<0 | 1>` — Controls whether the master can run the `kill` command on the clients in the background or the foreground.
Default: `0`

`0` Lets the master run the `kill` command run in the background. With this setting, the master kills the pending and running jobs on the clients at the beginning of the procedure and exits the tool.

`1` Lets the master run the `kill` command run in the foreground. When the parameter is set to `1` and the master is killed, the master runs the `kill` command for the clients and waits for the jobs to finish before exiting the tool.

bolt_liberate_error_flag_compatibility

`<0 | 1>` Controls how flows using the Bolt job distribution system handle errors. This parameter can be set at cell level.
Default: `0`

Note: This parameter only impacts the Bolt-related flows. The legacy Packet Arc or Packet Cell flows are not impacted.

`0` Errors that result in failed data following characterization will prevent LDB assembly for that cell or model generation for the entire LDB.

`1` Instructs the tool to ignore specific error messages regarding failed data. This will allow for assembly of partial LDBs and model generation for cells with failed data, but will also disable functionality that retries failed jobs during characterization.

Use recovery flows to re-characterize LDBs with failed data.

Example

```
set_var -cells abc* bolt_liberate_error_flag_compatibility 1
```

bolt_mpvt_scheduling

`<pvt_based | cell_based>`

Specifies how jobs are scheduled by the Bolt scheduler.
Default: `cell_based`

`cell_based` Ensures that the jobs are scheduled for all PVTs at the same time. The cells are sorted based on numerous criteria controlled by bolt_cell_priority_criteria.

`pvt_based` Ensures that the simulation jobs are submitted for the default PVT first because the default PVT needs to be characterized before any other PVT. If there are additional slots available, the Bolt scheduler starts submitting simulation jobs for the other PVTs also following the `cell_based` priority.

bolt_network_stats

`<0 | 1>` Controls writing of network statistics in the Liberate Trio server and client log files. The transactions per second and megabytes per second are recorded and displayed periodically.
Default: `0`

`0` Disables writing of network statistics.

`1` Enables writing of network statistics

When this parameter is set to `1`, the Liberate Trio server displays the total network traffic caused by all the jobs it sends to or receives from clients, and the traffic from itself to the Bolt server. The clients display the traffic that they generate themselves in isolation as they process the jobs and send heartbeats to the Bolt server.

bolt_post_char_command_distribution

<0 | 1>	Controls whether the distributed modeling flow should be enabled and skips the characterization process. Default: 0
0	Disables the distributed modeling flow.
1	Enables the distributed modeling flow.

The distributed modeling flow enables cell-level distribution of the tasks on the Bolt clients. The commands that support cell-level parallel distribution, such as compare_library, perform the cell-level comparison on clients as assembly jobs and create a cell-level result file. Once this is done, on the server side, the tool automatically appends/merges the cell-level results together to get a library-level result file. During this process, use the same characterization script that you used during the fresh characterization session along with the settings to enable the distributed modeling flow.

To enable the distributed modeling flow, before running the read_ldb command, set the bolt_post_char_command_distribution parameter to 1 that resets the multi_pvt_recovery_flow parameter setting to 0.

Example

```
set_var  bolt_post_char_command_distribution  1
set_var multi_pvt_recovery_flow 0
read_ldb LDB.ldb.gz
```

bolt_reuse_vectors_path

<string>	Specifies the path to the reuse vectors.

If a valid path is provided with the bolt_reuse_vectors_path parameter, Liberate checks for the vectorLdbs directory at the given path. If a vectorLdbs directory is found, then for all the cells that need to be characterized, links are created for all the vectorLdbs directory in the new run area.

If either the vectorLdbs directory or the result file for the cell is missing, then the preprocessing is done again for the job. Otherwise, the simulation jobs are submitted for the cell.

The specified path should be the LDB path, but not the one to the vectorLdbs directory. For example:

Correct Path:

```
set_var bolt_reuse_vectors_path /work/test/new_data/ldb.ldb.1.gz
```

Incorrect Path:

```
set_var bolt_reuse_vectors_path /work/test/new_data/ldb.ldb.1.gz/vectorLdbs
```

bolt_rsh_cmd_use_arrays

`<0 | 1>` Controls whether the job array support should be enabled for the bolt job distribution system.
Default: `1`

- `0` Disables job array support for the bolt job distribution system.
- `1` Enables job array support for the bolt job distribution system.

For examples of controlling the job arrays from the command line, see the Job Array Support for LSF and Sun Grid Engine subsection under the Bolt Job Distribution System section of Chapter 3, “Parallel Processing.”

This parameter must be used before the char_library command.

bolt_set_active_pvts

`<list>` Specifies a list of PVT names to characterize in a session out of all the available PVTs from the define_pvt command.

When the `bolt_set_active_pvts` parameter is set, Liberate prunes the list of the PVTs it has to characterize based on the user-provided list. If an invalid PVT name is provided, a warning message is displayed and the tool characterizes only those PVTs that are active.

bolt_use_durable_queues

`<0 | 1>` Controls the impact of restarting a Bolt server cluster or a single Bolt server on the Liberate Trio server.
Default: `0` (or `false`)

Note: This parameter is associated with the Bolt Job Distribution System. See also packet_arc_job_manager.

- `0` If a single Bolt server or the entire Bolt server cluster goes down, the Liberate Trio server session also restarts and does not resume from the point when the Bolt server went down.
- `1` When a single Bolt server or a Bolt server cluster is restarted, the Liberate Trio server session resumes from the point when the Bolt server went down.

When this parameter is set to `1`, it ensures that the messages from the message queues on the Bolt server are stored to the disk. If the server is taken down or it crashes, the stored messages are automatically reloaded when the Bolt server restarts. All Liberate Trio servers running at the time when the Bolt server went down resume from where they had left off. To know about how long the Liberate Trio servers will wait for the Bolt server to restart, you can check the settings for the following parameters:

- bolt_connection_lost_random_interval
- bolt_connection_retry_forever
- bolt_connection_retry_interval
- bolt_connection_retry_timeout
- bolt_connection_timeout

This parameter must be used before the `char_library` command.

bolt_zip_cell_log_files

`<0 | 1>` Controls whether the cell-level directories for log files should be tarred and zipped after the cell has been characterized.
Default: `0` (disabled)

`0`	Do not tar and zip the cell-level directories of log files.
`1`	Enables the feature to tar and zip the cell-level log files after the cell has been characterized and then delete the original log files. This setting is useful when packet_arc_job_manager=`bolt` because cell-level directories are created for the log files that in turn have separate log files for each job of that cell. This creates a lot of files.

When `bolt_zip_cell_log_files` is set to `1`, the zipped log files are named as `<`*`cellName`*`>_logs.gz` and saved in the `<`*`pvtName`*`>` directory. These files include a statement such as following to mark the cell: *Combined log file for cell: "cellName"*

In addition, the zipped files include the *"FILENAME: "* statements that mark the beginning of each individual job's log file. The files are placed in the order in which they get created.

bolt_zip_cell_log_files_on_forked_job

`<0 | 1>` Controls whether to disable zipping of the cell log files on the master and enable the zipping on the forked jobs.
Default: `0`

`0`	Disables the feature.
`1`	Zips the cell logs on the forked jobs and writes a log of the zipping activity in the log directory. Also, tests the existence of the zipped file and then deletes the cell-level log directory if the zipped file is present only on the client job.

Note: This parameter works when bolt_zip_cell_log_files is set to `1`.

bundle_arc_mode

`<0 | 1>` Specifies the method to choose the data to output in the bundle.
Default: `0`

- `0` If the `-use_pin` option is specified, it acquires the arc data from the specified use pin. If it is not specified, the data is acquired from the first bit of the bundle.
- `1` Liberate automatically selects the timing and power arcs from the characterized pins based on the user-defined criteria.

This parameter affects how the define_bundle_pins command chooses the pin data to output in the bundle.

This parameter must be set before the write_library command is run.

bus_syntax

`<string>` Specifies the characters that are used to delimit bus indexes from the bus name. The valid characters are: `"<>"`, `"()"`, or `"[ ]"`.
Default: `"[ ]"`

This parameter should only need to be set in conjunction with the `define_bus` command, if the input spice netlist or ldb do not use the default bus syntax. If the bus is defined using the `define_cell` command, then this parameter is set automatically. If buses are defined during library characterization, the bus_syntax is stored in the ldb and does not need to be reset when reading the ldb.

Note: This parameter does not control the `bus_syntax` used when writing the output library. That is controlled by the `-bus_syntax`. option of the `write_library` command.

Any pin in the ldb that ends with a number surrounded by the `bus_syntax` and belongs to a defined bus is outputted under a bus group in the Liberty file. Buses are defined either with the `define_bus` command or by the `-input`, `-output`, or `-inout` options of the `define_cell` command, using the following naming convention:

`<bus_name><bus_syntax_open><from_index>:<to_index><bus_syntax_close>"`

This parameter must be set before the write_library command is run.

Examples

Example1:

```
"dout[3:0]"
    or
"din<0:4>"
```

Example 2:

If you want to change the bus syntax in a library, then the following commands in same sequence will work. In this example the bus syntax is "<>" and the library is written with "[]".

```
set_var bus_syntax "<>"
read_library input.lib
write_library -bus_syntax "\[\]" -filename output.lib test1
```

bypass_read_invalid_ldb

<0 | 1>	Controls the expected behavior when an empty LDB directory is found or the LDB directory contains invalid files in the read_ldb-based recovery flow. Default: 0 Recommended: 1
0	Disables the distributed modeling flow.
1	When in the read_ldb-based recovery flow an empty LDB directory is found or the LDB directory contains invalid files, the tool continues to run the fresh characterization flow instead of generating an error and exiting the session.

Note: For single PVT and multi-PVT flows in Liberate Trio, this parameter is supported only when packet_clients>0 and packet_mode=arc.

Example

Example 1

When an empty LDB directory is found in the `read_ldb`-based recovery flow:

```
set_var bypass_read_invalid_ldb 1
read_ldb empty_dir.ldb.gz
char_library ...
```

Example 2

When the LDB directory contains invalid files for the `read_ldb`-based recovery flow:

```
set_var bypass_read_invalid_ldb 1
read_ldb invalid_files_in_dir.ldb.gz
char_library ...
```

capacitance_attr_mode

`<0 | 1>` Controls when the `rise_capacitance` and `fall_capacitance` attributes should be written separately. Default: `0`

- `0` Separate `rise_capacitance` and `fall_capacitance` attributes are written only when both capacitances are measured and are greater than `0`.
- `1` Separate `rise_capacitance` and `fall_capacitance` attributes are always written.

This parameter must be used before the `char_library` command.

capacitance_pin_rollup_k

`<value>` — User-defined multiplier is used in formula for calculating pin capacitance range. The value can be set between `0` to `1.0`. Default: `0.5`

You can also refer to capacitance_pin_rollup_mode.

This parameter must be used before the `char_library` command.

capacitance_pin_rollup_mode

`<0 | 1>` — Enables an alternate algorithm for calculating pin capacitance and pin capacitance range. Default: `0`

- `0` — Pin capacitance (`rise_capacitance` and `fall_capacitance`) is determined by the command set_pin_capacitance. Pin capacitance range is determined according to the setting of capacitance_range_mode.
- `1` — Pin capacitance is calculated as an average across all states, timing arcs. and output loads. See Calculating Pin Capacitance Range.

Calculating Pin Capacitance Range

Pin capacitance range `[C_nldm_min, C_nldm_max]` is computed as follows:

```
C_nldm_min = K * C_avg + (1-K) * Cmin
```

where: Cmin = min [Cpin(inpSlew, load, when)]

C_avg = avg[Cpin(inpSlew, load, when)]

- and -

`C_nldm_max = K * C_avg + (1-K) * Cmax`	where:	Cmax = max [Cpin(inpSlew, load, when)] C_avg = avg[Cpin(inpSlew, load, when)]

Here,
`"K"` is a user-defined multiplier set by the capacitance_pin_rollup_k parameter. The K-factor only changes the values in capacitance rise/fall *range* and has no effect on “capacitance” attribute itself.

Recommendations:

When using `capacitance_pin_rollup_mode`, set the pin capacitance "state" and "table" to average:

```
set_pin_capacitance -state avg -table avg
```

Also set the capacitance measurement thresholds to 0-100%

```
set_var measure_cap_lower_rise 0.0
set_var measure_cap_upper_rise 1.0
set_var measure_cap_lower_fall 0.0
set_var measure_cap_upper_fall 1.0
```

This parameter must be used before the `char_library` command.

capacitance_range_mode

`<0 | 1>` Controls how the capacitance range is calculated. Use the absolute min or max value or avg of min and avg of max.
Default: `0` (Use absolute min or max values.)

- `0` The lower/upper boundary is the min/max value of all the capacitances.
- `1` The lower boundary value is calculated as the average across all states, timing arcs, and output loads for the smallest input slew.

  The upper boundary value is calculated as the average across all states, timing arcs, and output loads for the largest input slew

This parameter must be set before the write_library command is run.

capacitance_save_mode

`<0 | 1>` Saves all pin capacitances.
Default: `0`
Recommended: `1`

- `0` Remove negative capacitances if `ccs_warn_negative_rcvr_caps = 1`. Keep negative capacitances if `ccs_warn_negative_rcvr_caps = 0`.
- `1` Saves all capacitances. Negative capacitances are saved as positive capacitances.

  If the tool-chain libraries are used, the recommendation is to keep negative receiver capacitances as this improves accuracy. To do this, leave the `capacitance_save_mode` parameter at the default value of `0` and set `ccs_warn_negative_rcvr_caps` to `0`.

Liberate derives capacitance values by integrating current. In cases where current direction is reversed from expectations, it is possible that capacitance values are negative. This condition is rare and usually is only seen on some pass-gate or tristate designs.

This parameter must be set before the `char_library` command.

ccs_abs_tol

<value>	Specifies the CCS absolute tolerance. Default: `1e-13`

Use this control parameter 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 parameter must be used before the `char_library` command.

ccs_base_curve_points

<value>	Specifies the number of base curve points. Default: `15`

Use this parameter to specify the number of base curve points used when generating compact CCS natively in Liberate. To output compact CCS format data, use the `write_library -ccs_compact` option.

This parameter can be used after the `char_library` command.

ccs_base_curve_share_mode

`<0 | 1 | 2>` Determines which algorithm is selected for reusing CCS base curves.
Default and recommended: `2`

- `0` Restores the behavior of 2.3p2 and prior releases.
- `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 parameter can be used after the `char_library` command.

ccs_cap_duplicate_risefall

`<-1 | 0 | 1 | 2>` Populates a missing rise or fall CCS receiver capacitance by duplicating the appropriate rise or fall CCS receiver capacitance from existing arc-based or pin-based data.
Default: `0` (copy from arc-based data)

Note: If a timing arc has a missing rise or fall CCS receiver capacitance, this parameter controls how Liberate handles various scenarios.

`-1` Leaves the missing group unpopulated. What is represented in the Liberty model is the same as what was characterized. It is recommended to use this value if all meaningful switching and hidden arcs are characterized.

`0` Copies the appropriate CCS receiver capacitance from the input pin involved in the timing arc.

Note: This default setting is a more accurate representation of the pin capacitance but might cause problems with downstream tools.

`1` Creates the missing rise or fall CCS receiver capacitance by duplicating the existing rise or fall CCS receiver capacitance.

2 Uses the Look Up Table (LUT) from the original table while copying a CCS receiver capacitance table. This is done to ensure that the dimension of the LUT and the cap table are the same.

Note: Two-sided arcs are functionally impossible in certain designs. Therefore, setting `-1` is the most accurate representation of such designs. Settings `0` and `1` were implemented to prevent Library Compiler and certain downstream tools from rejecting libraries that only have one-sided arcs, but are not as accurate as `-1` setting. If the tool-chain requires one-sided arcs to be fully populated, setting `0` is recommended.

This parameter must be set before the write_library command is run.

Examples

Example 1

```
ccs_cap_duplicate_risefall=-1
cell () {
    ...
    pin (Q) {
        timing () {
             # don't copy. receiver cap is one-sided.
             receiver_capacitance_rise (QR:template_2x2)
        }
    }
    pin (CP) {
        receiver_capacitance () {
             receiver_capacitance_rise (CPR:template_1x1)
             receiver_capacitance_fall (CPF:template_1x1)
        }
    }
    ...
}
```

Example 2

```
ccs_cap_duplicate_risefall=0
cell () {
    ...
    pin (Q) {
        timing () {
             # copy receiver_capcitance_fall from pin CP.
             # Create 2D table. index_2 is 0
             receiver_capacitance_rise (QR:template_2x2)
             receiver_capacitance_fall (CPF:template_2x2)
        }
    }
    pin (CP) {
           receiver_capacitance () {
             receiver_capacitance_rise (CPR:template_1x1)
             receiver_capacitance_fall (CPF:template_1x1)
           }
         }
```

```
            . . .
 }
```

3. ccs_cap_duplicate_risefall=1

```
cell () {
    ...
    pin (Q) {
      timing () {
       # copy receiver_capacitance_fall from receiver_capacitance_rise in same timing group.
        receiver_capacitance_rise (QR:template_2x2)
        receiver_capacitance_fall (QR:template_2x2)
      }
    }
    pin (CP) {
      receiver_capacitance () {
        receiver_capacitance_rise (CPR:template_1x1)
        receiver_capacitance_fall (CPF:template_1x1)
      }
    }
    ...
}
```

4. ccs_cap_duplicate_risefall=2

```
 cell () {
```

```
    ...
    pin (Q) {
      timing () {
        # copy receiver_capacitance fall from pin CP. keep 1D table
        receiver_capacitance_rise (QR:template_2x2)
        receiver_capacitance_fall (CPF:template_1x1)
      }
    }
    pin (CP) {
      receiver_capacitance () {
        receiver_capacitance_rise (CPR:template_1x1)
        receiver_capacitance_fall (CPF:template_1x1)
      }
    }
    ...
}
```

ccs_cap_enhancement_format_mode

`<0 | 1 | 2>` Enables the modeling of N-segment CCS receiver capacitance models.
Default: `0`

In 2016, the Liberty Technical Advisory Board (LTAB) ratified the N-segment CCS receiver capacitance tables. Use this parameter to enable creating libraries with N-piece models instead of 2-piece models.

`0`	Model 2-piece CCS receiver capacitance models.
`1`	Model N-piece CCS receiver capacitance models.
2	Model both 2-piece and N-piece CCS receiver capacitance models.

By default, Liberate models only the 2-piece CCS receiver capacitance. To model N-piece receiver capacitance, set the required threshold using the set_receiver_cap_thresholds command and ensure that the `ccs_cap_enhancement_format_mode` parameter is set before the write_library command. If the thresholds specified in the `set_receiver_cap_thresholds` command do not include the 2-piece thresholds (delay and slew), the list of thresholds will be augmented automatically.

Note: Starting from the LIBERATE 18.1 base release, if you need to model multiple flavors of N-segment CCS receiver capacitance models in the same session, ensure that each `ccs_cap_enhancement_format_mode` parameter setting is followed by a `write_library` command as shown in the example below:

```
set_var ccs_cap_enhancement_format_mode 0
write_library
set_var ccs_cap_enhancement_format_mode 1
write_library
set_var ccs_cap_enhancement_format_mode 2
write_library
```

ccs_cap_hidden_pin

`<0 | 1 |2>` Controls the output of the CCS receiver pin capacitance on inputs with hidden power arcs.
Default: `2` (Output pin capacitance for hidden arcs)

- `0` Uses the behavior of release 2.3p1 and prior where Liberate 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.

This parameter must be specified before the `char_library` command.

ccs_cap_hidden_pin_mode

`<0 | 1 >` Enables Liberate to store the CCS receiver capacitance data into the LDB for all `hidden_power` arcs.
Default: `0` (Output pin capacitance for hidden arcs)
Recommended: `1`

Note: The `-state` option of the set_pin_capacitance command specifies the method used to determine the capacitance in the default receiver capacitance under the pin.

- `0` Only stores the CCS receiver capacitance data for one of the characterized `hidden_power` arcs.
- `1` Stores the CCS receiver capacitance data for all characterized `hidden_power` arcs. (Recommended)

This parameter must be specified before the `char_library` command.

ccs_cap_is_propagating

`<0 | 1 | 2 | 3>` Enables output of the `is_propagating` attributes into the library.
Default: `1`

`0`	Do not output the `is_propagating` attribute.
`1`	Add the `is_propagating` attribute to pin-based `ccs_receiver` capacitance tables if an internal power group exists from the same pin with the same `when` condition where any CCB output is toggling.
`2`	Add the `is_propagating` attribute based on the characterized CCSN groups.
`3`	Use this setting when the ccsn_use_io_ccb_format parameter is set to `1` to annotate active input CCB groups on the receiver pin capacitance.

This parameter must be used before the `char_library` command.

ccs_cap_is_propagating_select_one_mode

`<0 | 1 >`

Requires Liberate to select and update at least one receiver capacitance-based active input CCB complex attribute with CCBs. It is related to the LC message (LBDB-712).
Default and recommended: `0`

Note: This parameter works with the IO CCB CCSN format (see ccsn_use_io_ccb_format =`1`). When using this format, set ccs_cap_is_propagating to `3` that populates the receiver capacitance-based active input CCB complex attributes with CCBs.

`0` Some pins might not contain an input CCB.

`1` Use at least one input CCB when available for receiver capacitance-based active input ccb modeling. Use this setting as a backup when CCSN characterization is requested, but none of the modeled receiver capacitances have any active input CCB constructs. This can happen when the receiver capacitances cannot be paired with any of the input CCBs under the pin, for instance when the receiver capacitances do not cause any CCB to toggle.

This parameter must be set before the write_library command is run.

ccs_cap_mode_add_missing

`<0 | 1 >`

Specifies whether to get 2.4-style libraries with missing receiver capacitance for one-sided timing arcs.
Default: `1`

`0` Gets 2.4-style libraries with missing receiver capacitance for one-sided timing arcs

`1` Gets 2.4-style libraries without missing receiver capacitance for one-sided timing arcs.

This parameter must be set before the write_library command is run.

ccs_cap_non_hidden_pin

`<0 | 1 >`

Controls the output of the CCS receiver pin capacitance on inputs without hidden power arcs. The output CCS receiver capacitance is copied from output pins and the capacitance table is converted from 2D to 1D. If there are multiple conditional receiver capacitance tables, the copy follows the set_default_group setting.
Default: `0`

`0`	Do not output CCS receiver capacitance on input pins without hidden power arcs.
`1`	Output CCS receiver capacitance on input pins without hidden power arcs.

This parameter must be set before the write_library command is run.

Example

Consider an inverter having an output pin (`Y`) that has arc-based `receiver_cap` groups such as following:

```
pin (Y) {
    timing () {
        receiver_capacitance1_rise (delay_template_5x5) {
            index_1 ("0.25, 1.5");
            index_2 ("0.015, 0.6");
            values ( \
                  "0.00666895, 0.00643509", \
                  "0.00737907, 0.00650522" \
                  );
        }
        receiver_capacitance2_rise (delay_template_5x5) {
            index_1 ("0.25, 1.5");
            index_2 ("0.015, 0.6");
            values ( \
                  "0.0104799, 0.00684902", \
                  "0.0133222, 0.0074708" \
                  );
        }
   }
```

```
}
```

If you set the `ccs_cap_non_hidden_pin` parameter to `1` and the `set_default_group -criteria` command option to `max`, the copied pin-based receiver cap table for input pin (`A`) look the following:

```
pin (A) {
    receiver_capacitance () {
        receiver_capacitance1_rise (receiver_cap_power_template_5x5) {
            index_1 ("0.25, 1.5");
            values ( \
                  "0.00666895, 0.00737907", \
                  );
        }
        receiver_capacitance2_rise (receiver_cap_power_template_5x5) {
            index_1 ("0.25, 1.5");
            values ( \
                  "0.0104799, 0.0133222", \
                  );
        }
    }
}
```

ccs_cap_use_input_transition

`<0 | 1 | 2>` Determines the pin transition direction followed to determine the CCS receiver capacitance (`receiver_cap`) direction. Default and recommended: `2`

- `0` Specifies to use the *output* pin transition direction for the `receiver_cap` direction.

  Restores the old behavior where the CCS receiver capacitance follows the *output* pin direction.

  Note: Set this parameter to `0` after `read_ldb` or `char_library` and before `write_library`.

- `1` Specifies to use the *input* pin transition direction for the `receiver_cap` direction during modeling.

2	Specifies to use the *input* pin transition direction for the receiver_cap direction during characterization. The enabled algorithm improves the accuracy of index_1 in the CCS receiver capacitance when the input waveform has an unusual shape and the input slew thresholds are non-standard. The input waveform can have an unusual shape, for example, when an active driver is used (see set_driver_cell). The standard input slew thresholds are 10/90, 20/80, and 30/70 with a delay threshold at 50% of supply.

This parameter must be set before the char_library command.

The read_library command resets this parameter to 0. The char_library and read_ldb commands set this parameter to 1.

If ccs_cap_use_input_transition and ccs_cap_hidden_pin is set to 1, then for one sided arcs such as "clock rising_edge to Q", the inactive edge (falling) uses the CCS receiver capacitance from the clock pin. The load index for this entry (index_2) is set to "0" because this data has no load dependency.

ccs_cap_use_input_transition_tristate

<0 | 1 >

Specifies that the CCS receiver capacitance for tristate arcs follows the input pin direction.
Default: 1

0	No check is performed to ensure that the CCS receiver capacitance for tristate arcs follows the input pin direction.
1	Checks that the CCS receiver capacitance for tristate arcs follows the input pin direction.

This parameter must be used before the char_library command.

ccs_correct_current_by_area

`<0 | 1 >`

Applies an alternate smoothing method that tracks the total area of current to be conserved during the smoothing process, if `ccs_waveform_smooth_mode` is set to `1`. Do not use with `ccs_smooth_lower_rise` or `ccs_smooth_upper_fall`.
Default: `0 (disabled)`

- `0` — Disables correct method.
- `1` — Enables correct current by area method for all simplified waveforms.

This parameter must be used before the `char_library` command.

ccs_current_model_pin_load

`<0 | 1 >`

Controls whether 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: `0`

- `0` — Instructs Liberate to not to characterize and model CCS current if the output has a parasitic load applied.
- `1` — Instructs Liberate to measure and report the CCS current even if a `define_pin_load` has been run.

This parameter must be set prior to the `char_library` command.

ccs_enable_sawtooth_out

`<0 | 1>` Enables the detection and correction of CCS with double-peaks current when smoothing is enabled.
Default: `0`

`0`	Does not detect double-peak (sawtooth) CCS current waveform data.
`1`	Detects double-peak CCS current waveform data and applies smoothing to the CCS current waveform.

This parameter must be used before the `char_library` command.

ccs_force_grid_delay

`<0 | 1 | 2>` Controls 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 parameter must be used before the `char_library` command.

ccs_from_ecsm_linear_interp_factor

`<value>`

Enhances the ECSM to CCS conversion process. This parameter adds linear points before the voltage curve fitting to avoid unusual time step that causes the curve fitting to fail or accuracy issues if the following condition is met:

```
ecsm time step >  ccs_from_ecsm_linear_interp_factor *
(trans/(measure_slew_upper_threshold -
measure_slew_lower_threshold))
```

For example, if the user-defined slew threshold is `0.25~0.75`, then the tool will get `ecsm time step > ccs_from_ecsm_linear_interp_factor *(trans/0.5)`, and perform linear interpolation.

This parameter must be used before the `char_library` command.

ccs_from_ecsm_smooth_mode

`<0 | 1 | 2>`

Checks if there is a large time step with miller voltage change. If found, the sharp segmentation is smoothened.
Default: `0`

`0`	The check is disabled.
`1`	Smoothens sharp segmentation only.
`2`	Recreates the waveform based on ecsm_13pts + delay + upper + lower threshold list.

This parameter must be used before the `char_library` command.

ccs_infer_output_dir

`<0 | 1>` Detects whether all outputs have both rising and falling waveforms.
Default: `1`

- `0` Address CCST issues with pulse generators having rising and falling waveforms in every output.
- `1` Detect whether all outputs have both rising and falling waveforms.

This parameter must be used before the `char_library` command.

ccs_init_voltage_comp_thresh

`<value>` Sets a voltage threshold to enable a proprietary initial voltage offset. Default: `1.1` (fully-compensated to improve correlation for PrimeTime 2010.06 and prior releases)

Set this parameter to `-1` to disable the compensation for improving correlation for PrimeTime 2010.12 and later releases.

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), the resulting CCS waveform may have an additional time delay and might not reach the final voltage rail. Liberate compensates for the non-rail initial voltage to ensure the CCS waveform matches the NLDM values and reaches the final rail voltage.

This parameter represents a percentage of the full-rail swing over which compensation takes place. If set to a value larger than 1, the maximum threshold for the compensation is controlled by the `measure_slew_lower_rise` and `measure_slew_upper_fall` parameters. If set to a value between 0 and 1, Liberate compensates to the value given. If set to a value less than 0, compensation is disabled.

The NLDM and CCS models are currently limited in how to handle non-rail starting or ending voltage levels, since `output_signal_level` attributes must be applied at the pin level. In cases where non-rail behavior is only observed for specific `WHEN` conditions, the accuracy of those models is degraded. See `ecsm_measure_output_range` for accurate modeling of these effects.

This parameter must be set before the `char_library` command.

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, Liberate issues a warning.

This parameter must be used before the `char_library` command.

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, Liberate stops adding points to the CCS data when the number of points reaches the limit specified by this parameter.

This parameter must be used before the `char_library` command.

ccs_multiple_switching_output_mode

`<0 | 1>` — Enables handling of multiple switching output voltage.
Default: `0`

- `0` — The output waveform does not have special processing.
- `1` — Filters a multiple switching output voltage waveform. Keeps the first or last rise or fall transition waveform according to the setting of the `measure_target_occurrence` parameter.

When the `simultaneous_switch` parameter is enabled and is greater than `0`, the output voltage behavior can be a multiple switching waveform (that is a waveform with multiple rise or fall edges). This can produce undesirable negative and/or positive current values in the rise or fall waveform. Liberate uses a fixed value to make the CCS library data LC friendly. This could lead to NLDM versus CCS comparison (`compare_ccs_nldm`) failures.

This parameter must be used before the `char_library` command.

ccs_process_post_ldb

`<0 | 1>` Requests that the original CCS waveform should be stored in the LDB.
Default: `0`

- `0` Process the CCS waveform data during characterization. The modified waveform is stored in the LDB.
- `1` Process the CCS waveform data during model generation. The original current waveform data in stored in the LDB.

This parameter must be used before the `char_library` command.

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, Liberate 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 parameter must be used before the `char_library` command.

ccs_segmentation_effort

`<-1 | 0 | 1 | 2 | 3 | 4 | 5>`

Controls the selection of the method used to acquire the CCS current waveform from simulated `I(t)` data.
Default: `1`

`-1`	Enables a basic waveform simplification method that skips all waveform manipulation except for compression.
`0`	Use minimal waveform processing to conform to the CCS Timing Library Characterization Guidelines version 3.3. For example, current cannot change directions.
`1`	Use the smoothed algorithm.
`2`	Use the dv/dt-based algorithm.
`3`	Use both 1 and 2, but choose the waveform that best correlates to NLDM.
`4`	Same as 1, but also applies CCS retry criteria (ccs_retry_mode) to check the reconstructed waveform instead of the original current waveform.
`5`	Specifies to measure voltage, including multi-crossed, for the specified threshold list (that includes both NLDM and ECSM measure thresholds) and derive it to get the CCS currents. **Note:** In Liberate LV and other STA tools, this setting ensures that the CCS voltage is calculated through `(volt_val2 = volt_val1 + c2/cap*(t2-t1))`, and then the compare function is run.

This parameter must be set before CCS waveform processing is performed (see ccs_process_post_ldb).

ccs_simplify_thresh_mode

`<0 | 1 | 2>` Specifies the method used to force storing of different types of data when the CCS simplify algorithm is run.
Default: `0`

`0`	Do not force storing of points when the CCS simplify algorithm is run.
`1`	Force storing of voltage delay_out_rise/fall, measure_slew_lower_rise/fall, measure_slew_upper_rise/fall points when the CCS simplify algorithm is run.
`2`	Force storing of ECSM list thresholds when the CCS simplify algorithm is run. This mode is enabled by default when ccs_segmentation_effort is set to `-1`.

This parameter must be used before the `char_library` command.

ccs_smooth_lower_rise

`<value>` Specifies a lower rise threshold to enable current waveform smoothing for rising outputs.
Default: `-1` (Disabled)

Sometimes, the simulator reports a severely non-monotonic CCS current waveform. If this *distorted* waveform is provided to the timer, it might cause significant timing differences. To avoid this error in the timer, Liberate can smooth the current waveform. To enable this smoothing, set the rise and fall thresholds by using the `ccs_smooth_lower_rise` and `ccs_smooth_upper_fall` parameters, respectively. A higher `ccs_smooth_lower_rise` value corresponds to more aggressive smoothing. To minimize the introduction of errors, the specified value should be equal to or less than the `measure_slew_lower_rise` threshold.

When a value is set for this parameter, Liberate checks if the CCS current waveform generated by the simulator has multiple peaks, where the initial peak is greater than the second peak. When detected, Liberate applies a proprietary smoothing algorithm to smooth the initial current peak to ensure the waveform meets the Liberty syntax rules while maintaining the original voltage/time on the output waveform at the specified threshold.

Ensure this parameter is used along with the corresponding fall threshold set by using the `ccs_smooth_upper_fall` parameter.

Note: The smoothing algorithm can have an impact on downstream timing or noise tools that are sensitive to the specified threshold.

This parameter must be used before the `char_library` command.

Example

```
set_var ccs_smooth_lower_rise 0.2
```

ccs_smooth_upper_fall

<value>	Specifies a upper fall threshold to enable current waveform smoothing for falling outputs. Default: `-1` (Disabled)

Sometimes, the simulator reports a severely non-monotonic CCS current waveform. If this *distorted* waveform is provided to the timer, it might cause significant timing differences. To avoid this error in the timer, Liberate can smooth the current waveform. To enable this smoothing, set the rise and fall thresholds by using the `ccs_smooth_lower_rise` and `ccs_smooth_upper_fall` parameters, respectively. A lower `ccs_smooth_upper_fall` value corresponds to more aggressive smoothing. To minimize the introduction of error, the specified value should be equal to or greater than the `measure_slew_upper_fall` threshold.

When a value is set for this parameter, Liberate checks if the CCS current waveform generated by the simulator has multiple peaks, where the initial peak is greater than the second peak. When detected, Liberate applies a proprietary smoothing algorithm to smooth the initial current peak to ensure the waveform meets the Liberty syntax rules while maintaining the original voltage/time on the output waveform at the specified threshold.

Ensure this parameter is used along with the corresponding rise threshold set by using the `ccs_smooth_lower_rise` parameter.

Note: The smoothing algorithm can have an impact on downstream timing or noise tools that are sensitive to the specified threshold.

This parameter must be used before the `char_library` command.

Example

```
set_var ccs_smooth_upper_fall 0.8
```

ccs_voltage_smooth_thresh

<voltage | -1> Sets a voltage threshold (for example, `0.2`) to enable a proprietary smoothing algorithm.
Default and recommended: `-1` (Disabled)

Note: The smoothing algorithm may have an impact on downstream timing or noise tools that are sensitive to that threshold.

`voltage`	Enables the smoothing algorithm. A higher number corresponds a more aggressive smoothing.
`-1`	Disables the smoothing

To reduce CCS interpolation error for some corner cases, Liberate can artificially smoothen the waveform.

This parameter must be set before the write_library 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 4.5% 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).

If CCS vector pad fails to reach the stopping point that `ccs_voltage_tail_tol` sets, by default no message is displayed. You can enable display of a warning message by using the set_message command as shown below:

```
set_message -id  477 -msg_limit 2
```

A message such as following will then be displayed:

```
WARNING (LIB-477): (CCS): The final signal voltage of the output_current_rise for
cell 'AND', pin 'Y', rel_pin 'A', when '!B', slew '2.82497e-12', load '1.83394e-
16' could not reach 0.77v(ccs_voltage_tail_tol*vswing+gndval = 0.951*0.810 +
0.000). Save and review the SPICE decks (see deck -), decrease
'ccs_voltage_tail_tol', enable measure_output_range and/or increase the
sim_duration and rerun.
```

This parameter must be used before the `char_library` command.

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 parameter 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`.

This parameter must be used before the `char_library` command.

ccs_voltage_tail_trim_tol

<value>	The stopping point as a ratio of supply of the CCS waveform during simulation waveform capture. Default: `0.999` (within 0.1% of rail supply voltage)

For CCS timing waveform data, the tail of the integrated `v(t)` obtained from the sampled `i(t)` is captured until `${ccs_voltage_tail_trim_tol} x supply_swing`, is reached.

This parameter controls the current waveform capture and works with `ccs_voltage_tail_tol` to control the modeling of the CCS current waveforms. For `28nm` processes and below, it is strongly recommended to set this value higher than `ccs_voltage_tail_tol`, but not to `100%`. This helps to remove `"N-curve"` phenomena in the CCS current waveforms.

The default setting should only be used for large geometries where capturing to within a tighter range creates such a large waveform that in time duration causes problems in legacy downstream tools.

This parameter must be used before the `char_library` command.

Example

```
set_var ccs_voltage_tail_trim_tol 0.999
```

ccs_voltage_waveform_style

`<compact | vector>`	Specifies the CCS voltage waveform data format to write into the output library. Default: `compact` **Note:** Use the `write_library -ccs_voltage_waveform` option to enable writing out the CCS voltage waveforms.
`compact`	Writes the compact form of the voltage waveform for ccs voltage data.
`vector`	Writes the vector format of the ccs voltage waveform.

This parameter must be set before the write_library command is run.

ccs_warn_negative_rcvr_caps

`<0 | 1 | 2>` Controls the type of data that are checked for handling of negative receiver capacitances.
Default: `0`

`0`	Disables the check.
`1`	Check both nominal and sensitivity.
`2`	Only check nominal or Liberate capacitance.

Liberate derives capacitance values by integrating current. In cases where current direction is reversed from expectations, it is possible that capacitance values are negative. This condition is rare and is usually only seen on some pass-gate or tristate designs.

If the tool-chain libraries are used, the recommendation is to keep negative receiver capacitances because this will improve accuracy. In order to do this, leave `capacitance_save_mode` at its default value of `0` and set `ccs_warn_negative_rcvr_caps` to `0`.

This parameter must be set before the write_library command is run.

ccs_waveform_min_time_step

<value> Specifies the minimum time step supported in the `ccs_waveform` (`index_3`). This is used when adjusting time for monotonicity.
Default: `1e-16` (in seconds)

This parameter must be set before the write_library command is run.

ccs_waveform_smooth_mode

`<0 | 1>` Enables an algorithm to smooth non-monotonic voltages resulting from the `output_current_*` waveforms. Do not use this parameter with `ccs_smooth_lower_rise` or `ccs_smooth_upper_fall`.
Default: `0` (disabled)

`0`	Disables ccs smooth method.
`1`	Enables new ccs smooth method.

This parameter must be used before the char_library command.

ccsn_allow_duplicate_condition

`<0 | 1 >` Allows multiple CCSN groups to be written to the library. Default: `0`

- `0` Keep only the worst-case group, and do not include the rest.
- `1` Allow multiple CCSN groups to be written to library.

Sometimes CCSN characterization results in having multiple ccsn groups with the same WHEN condition (e.g. internal nodes might be initialized differently and are not captured in the WHEN condition). By default these groups are ***not*** written into the library.

Note: We recommend setting this parameter consistent with the requirements of the noise analysis tool that will be using the library. If the noise analysis tool does not support duplicate conditions, then set this parameter to `0`.

This parameter must be used before the char_library command.

ccsn_allow_multiple_input_switching

`<0 | 1 | 2>` Allows the vectors on the various input pins on a Channel Connected Region (CCR) to switch independently. Default and recommended: `2`

- `0` Permits only the CCB input to switch. Use this setting when the input pins to the CCR cannot switch independently.
- `1` Allows the input pins to the CCR to switch independently. Use this setting while characterizing CCRs that only propagate noise when the inputs can be toggled in opposite directions independently.
- `2` Enforces the same functionality as setting of `1`, but with improved recognition of multiple independent input-switching events to the CCR.

This parameter must be used before the char_library command.

ccsn_allow_overlap_when

`<0 | 1 | 2>` Controls the output of arc-based CCSN data.
Default: `1`

Use this parameter to resolve an issue where vector data is not properly matched up to the corresponding timing groups for arc-based CCSN generation. This occurs only in a verbose template-based flow because the timing group can have a nonempty `when` string while the vector data has a null string for the `when` condition.

- `0` Restores the behavior of the 2.3 and earlier release that check for an exact match of the `when` condition.
- `1` Checks for a Boolean overlap instead of requiring an exact `when` string match.
- `2` First checks for an exact match of the `when` string as done in the setting of `0`. Then, if no match is found, checks for a Boolean overlap as done in the setting of `1`. The setting of `2` allows the exact match to take precedence over a partial match.

This parameter must be used before the char_library command.

ccsn_allow_partial_voltage_swing

`<0 | 1 >` Allows CCSN output_voltage groups to use partial voltage swings instead of the values recommended in the CCSN Characterization Guideline.
Default: `1`

- `1` Liberate overrides the recommendations with the voltage levels measured in the transition. Where possible, the voltage levels from the recommendations are used. If necessary, indexes and values are padded to insure monotonicity

By default, Liberate uses the voltages recommended in the CCSN Characterization Guidelines for output_voltage_* groups. These are `10%`, `30%`, `50%`, `70%`, and `90%` of the rail voltage. In certain cases, cells are unable to meet these voltage levels usually due to large leakage current. In these cases, Liberate issues a warning during characterization and a warning during modeling. These CCSN groups are not written out in the library.

This parameter must be used before the char_library command.

ccsn_arc_channel_check

`<0 | 1 | 3 | 4>` Aggressively prunes non-channel connected CCSN stages that might end up as arc-level CCSN constructs by enabling a more thorough vector dependent check.
Default: `1`

- `0` Enables pruning.
- `1` Disables pruning, that is, restores the behavior of 3.0p3 and prior releases.
- `3` Improved CCSN partitioning for pass-gate structures.
- `4` Enables the check for transmission gate when performing CCSN partition. Set `ccsn_arc_channel_check` to `4` to handle checks dealing with multiple transmission gates in a series.

This parameter must be used before the char_library command.

ccsn_arc_consistent_cut

`<0 | 1 >` Uses an enhanced algorithm to ensure that a consistent cut node is chosen for arc based CCSN. CCSN generation in Liberate chooses a single stage CCSN arc over a two stage CCSN arc, whenever possible.
Default: `1`

- `0` Restore the behavior of 2.4p2 and prior releases.
- `1` Enable an enhanced CCS cut algorithm.

This parameter must be used before the char_library command.

ccsn_arc_high_effort

`<0 | 1 | 2>` Control output of arc-based CCSN data.
Default: `1`

- `0` Restore the pre 2.2p2 behavior, which does not output state-dependent CCSN arcs.
- `1` Enable the output of state-dependent CCSN arcs.
- `2` Same as `1`, but also outputs a message in the log file for each cell that triggers a state-dependent arc code.

Liberate outputs more arc-based CCSN stages by applying flexible side input requirements. The result is that some types of cells, such as MUXes, have an increase in the number of arc-based CCSN data. This control parameter can be used to restore the behavior prior to 2.2p2 release.

This parameter must be used before the char_library command.

Example:

```
set_var ccsn_arc_high_effort 0
```

ccsn_bus_holder_mode

`<0 | 1 | 2>` Specifies methodology for modeling unbuffered output or inout pins.
Default and recommended: `2`

`0`	Do not model unbuffered outputs. (Behavior of release 3.2p2 or earlier.)
`1`	Cleanup first stage only; do not add a last stage.
`2`	Cleanup feedback in first stage (on pins having `bidi` ports) and add a last stage.

This parameter must be used before the char_library command is run.

ccsn_channel_inputs_high_effort

`<0 | 1 >` Controls merging of CCSN data.
Default: `1`

`0`	Restore the behavior of 2.3 and prior releases.
`1`	Enable CCSN constructs for unbuffered input pins.

This parameter must be used before the char_library command is run.

ccsn_check_data

`<0 | 1>` Controls how to perform CCSN static checks including DC current, propagated noise waveform and output voltage.
Default: `1`

- `0` Only perform the checks and do not remove invalid CCSN group. You can review all associated messages in the log file.
- `1` Perform the checks and remove the invalid CCSN group.

This parameter must be used before the write_library command is run.

ccsn_check_data_max_dc_current_min_thresh

<value> Checks for CCSN DC currents greater than the specified threshold. If the maximum CCSN DC current is less than the threshold, Liberate issues an associated message.
Default: `1e-6A (1uA)`

This parameter must be used before the write_library command is run.

ccsn_check_dc_tables

`<0 | 1>` Controls whether monotonicity check should be enabled on the DC tables when the `write_library` command is run.
Default: `0` (`false`)

- `0` Disables monotonicity check on the DC tables.

  Note: Alternatively, you can specify the value as `false`.

1	Enables monotonicity check on the DC tables and fixes non-monotonicity, if needed. However, for the latter to happen, you must set the write_library -fix_dc option. The non-monotonicity is fixed using a pre-loaded training database that contains normalized 'good-shaped' DC tables. For more information, see read_training_data, write_training_data, and update_training_data. **Note:** Alternatively, you can specify the value as true.

This parameter must be used before the write_library command is run.

Examples

1. DC table checks (generate report only, without any fixes):

```
set_var ccsn_check_dc_tables true
read_library $lib
write_library … -ccsn $lib
```

2. DC table fixes:

```
read_training_data $dbList
read_library $lib
write_library … -ccsn -fix_dc $lib
```

3. Generate a new training database / merge existing databases:

```
# Enable to read both training databases and pure .lib files.
read_training_data $dbList ;
# Liberate reads and checks the data, and then creates one new database.
write_training_data $newDb
```

4. Update the training database on the fly while checking the write_library data:

```
read_training_data $dbList
set_var update_training_data true
read_library $lib
write_library … -ccsn -fix_dc $lib
write_training_data $newDb
```

ccsn_check_non_peak_noise_prop_range

`<0 | 1>` Checks whether the fourth and fifth values in propagated noise tables are within appropriate signal range for one-sided stage CCSN groups.
Default: `1`

- `0` Enables backward compatibility with LIBERATE 17.1 ISR1.
- `1` Enables the check.

This parameter must be used before the char_library command is run.

ccsn_consistent_side_inputs

`<0 | 1 >` Enables to avoid an unexpected termination of the process that can occur when there are CP->Q arcs with no **when** conditions but different internal side-input values.
Default: `1`

- `0` Restore the behavior of 2.3 and earlier releases.
- `1` Avoids an unexpected termination of the process.

This parameter must be used before the char_library command is run.

ccsn_controlling_path_check

`<0 | 1 >` Specifies how to handle CCSN output nodes that are driven by multiple paths including pass gates.
Default: `0`

- `0` Apply default algorithm.

1	Check the CCSN CCCs for the presence of multiple active paths that could affect the output pin. When found, Liberate preserves the primary active path from the related pin to the CCSN CCC output while disabling secondary paths. Liberate identifies and modifies the controlling signals at the pass gates along the secondary paths. If such a CCC is identified at the arc level, the CCSN group is skipped and a CCSN group is identified at the pin-level instead that can be modified as previously described.

This may occur when the CCC output probe wire is controlled by pass gates as well as normal driver stages. In addition, some of the pass gates are controlled by the related pin directly (via the same CCC) or indirectly by other signals. Turning them `OFF` or `ON` completely as a part of the same CCSN stage is problematic.

This parameter must be used before the char_library command is run.

ccsn_dc_static_check

`<0 | 2 | 3>` Performs a check to determine if the CCSN partition is valid.
Default: `0`
Recommended: 3

`0`	Do not perform any data checks on CCSN DC tables.
`2`	Apply basic data checks on `ccsn_first_stage` DC tables.
`3`	Apply a data check by walking across the DC table and comparing the values at 0 and VDD. A potential error is flagged if the values indicate that the CCSN partition has not been "turned on", that is, the values must show a sign change, or magnitude difference of 10 or greater to be considered a valid CCSN partition. For more information, see ccsn_dc_static_check_thresh.

This parameter must be set before the write_library command is run.

ccsn_dc_static_check_mode

`<0 | 1 >` Controls severity of messaging when a problem in the CCSN data is found.
Default: `1`

`0`	Report a warning when a `ccsn_dc_static_check` is enabled and a problem in the data is found.
`1`	Report an error when a `ccsn_dc_static_check` is enabled and a problem in the data is found. In addition, remove the problem CCSN data table from the output library.

This parameter must be set before the write_library command is run.

ccsn_dc_static_check_thresh

<value>	Specifies the threshold applied when `ccsn_dc_static_check=3`. Set this to the desired threshold that identifies a turn on or a non-static behavior in a CCSN DC table. Default: `10`

This parameter must be set before the write_library command is run.

ccsn_dc_template_size

<value>	Specifies the size of the CCSN DC current table to be used. Default and recommended: `29`

For larger geometries, this parameter can be set to `17` to allow for better characterization runtime. For advanced nodes or if using a version of PrimeTime that is 2012.x or later.

This parameter must be used before the `char_library` command.

ccsn_default_conditional_check

`<0 | 1>`	Controls how various pull-up/pull-down capable vectors are analyzed to determine if two sided stages are possible. Default (and Recommended): `1`
`0`	Ignore default timing group related vectors.
`1`	Include vectors related to default timing groups when checking for two sided stages.

This parameter must be used before the char_library command is run.

ccsn_default_group_add_when

`<0 | 1 >` Adds the *when* condition inside default CCSN groups.
Default: `0` (Do not add WHEN.)

- `0` Does not add the *when* condition inside the default CCSN groups.
- `1` Add the *when* condition inside the default CCSN groups.

This parameter must be used before the char_library command is run.

ccsn_default_group_criteria_mode

`<0 | 1 >` Enables generation of worst case CCSN default groups at the pin-level and at the default timing group.
Default and recommendedt: `0`

- `0` Generate libraries that are structurally independent of per-corner data.

  Important

  For libraries to be used in voltage or temperature scaling flows, we highly recommended setting this parameter to 0
- `1` Generate the most conservative per-corner libraries.

This parameter has effect only if the library has timing arcs without CCSN groups.

Note: This may result in multiple default CCSN groups, depending on the data generated during characterization.

This parameter must be used before the char_library command is run.

ccsn_dual_tie_enable

`<0 | 1 | 2>` Enables CCSN modeling for tie-high and tie-low cells.
Default (and Recommended): `2`

- `0` Restores the behavior of 2.5 and prior releases where CCSN data was not output for tie-high or tie-low cells.
- `1` Enables CCSN modeling for tie-high and tie-low cells.
- `2` Enables enhanced CCSN modeling for dual tie cells. Setting of this value is recommended when:
  - tie-high and tie-low pin do not have the first stage
  - tie-high and tie-low pin output_ccb group has only static dc_current table

This parameter must be used before the char_library command is run.

ccsn_extra_default_stages

`<0 | 1 | 2>` Controls merging of CCSN data.
Default: `1`

Use this parameter to determine when Liberate will add default (state independent) CCSN data under the input pin.

- `0` Do not add default CCSN groups under the input pin when all the CCSN stages are at arc level.
- `1` Check each timing group for the existence of CCSN data. Add default pin-level CCSN data for the pin and related pin for any timing group that has no CCSN data.

`2` Liberate will try to add default CCSN data to all cell pins. Liberate will copy all arc-based CCSN stages to pin level as default CCSN group.

Note: This can result in multiple default CCSN entries under the input pin.

This parameter must be used before the model creation starts with the write_library command.

ccsn_filter_probe_mode

`<0 | 1>` Controls whether additional checks should be performed while filtering CCSN stage outputs.
Default: `1`

Note: This parameter can be set at cell level.

`0` Do not perform any additional checks.

`1` Conduct additional checks while filtering CCSN stage outputs.

A wire is considered as a valid probe wire if at least one of the following conditions is met:

- Connects to NMOS (discharge path) AND PMOS (charge path)
- Drives non-dummy CCC

ccsn_first_stage_load1

<value>

This parameter (denoted as C1 in the Liberty documentation) allows you to specify the load to be used on the output of a `ccsn_first_stage` when characterizing CCSN.

In the section titled *CCS Noise Propagation Tables* of the CCSN Characterization guideline, this parameter corresponds to load capacitance C1.

Default: `-1`

`-1`
When there are multiple CCB stages, no load is added to the stage output.

When there is only one single CCB stage (such as a NAND gate), Liberate will try to retrieve the maximum load by `index_2`. The `ccsn_first_stage_load1` will be set to `"max_load*0.05"` (5% of the max load).

<value>
Specifies the load that needs to be applied to the output of the `ccsn_first_stage`. (Unit=Farad)

This parameter must be used before the char_library command is run.

ccsn_first_stage_load2

<value>

This parameter (denoted as C2 in the Liberty documentation) allows you to specify the load to be used on the output of a `ccsn_first_stage` when characterizing CCSN.

In the section titled *CCS Noise Propagation Tables* of the CCSN Characterization guideline, this parameter corresponds to load capacitance C2.

Default: `-1`

`-1`	When there are multiple CCB stages, no load is added to the stage output. When there is only one single CCB stage (such as a NAND gate), Liberate will try to retrieve the maximum load by `index_2`. The `ccsn_first_stage_load2` will be set to `"max_load*0.10"` (10% of the max load).
`<value>`	Specifies the load that needs to be applied to the output of the `ccsn_first_stage`. (Unit=Farad)

This parameter must be used before the char_library command is run.

ccsn_floating_init_mode

`<0 | 1 | 2 | 3>` Controls how Liberate initializes floating internal cell simulation wires in the Channel Connected Block (CCB) for CCSN simulation. These are nodes that are internal to the cell (not primary ports).
Default: `1` (use `.nodeset`)
Recommended: `3`

`0`	Uses `.ic` in all CCSN simulations. This setting is used for backward compatibility for 12.1 ISR3 and prior releases.
`1`	Uses `.nodeset` in all CCSN simulations.
`2`	Uses `.nodeset` in CCSN DC simulations and `.ic` in CCSN transient simulations.
`3`	Uses `.nodeset` in all CCSN simulations except dc_current simulations which have no `.nodeset` or `.ic`. The simulator will need to initialize all nodes for the dc_current simulations.

This parameter must be used before the char_library command is run.

ccsn_include_passgate_attr

`<0 | 1 >` Enables modeling of pass_gate attributes.
Default (and Recommended): `1`

- `0` Does not automatically model these pass gate related attributes.
- `1` Automatically model these pass gate related attributes.

The Liberty syntax supports the following pin-level attributes for unbuffered output latches:

- `is_unbuffered`: Indicates that a pin is unbuffered.
- `has_pass_gate`: Indicates whether the pin is internally connected to at least one pass gate.
- `is_pass_gate`: Indicates whether the `ccsn_*_stage` data is modeled for a pass gate and is specified in a `ccsn_*_stage` group (such as `ccsn_first_stage`).

This parameter must be used before the char_library command is run.

ccsn_io_allow_multiples

`<0 | 1 >` Enables exhaustive CCSN partition analysis when the IO (port-based) partitioning algorithm is used.
Default: `1`

- `0` Liberate constructs a CCSN partition CCC using a heuristic based algorithm.
- `1` Liberate constructs and analyzes all possible CCSN partitions when the IO partitioning algorithm is used. This setting increases the runtime. You can use this setting when a more detailed IO partition based CCSN analysis is desired

The generation of CCSN data requires special circuit partitions. When the *Inside View* algorithm is unable to sensitize a Channel Connected Component/Region (CCC) or is disabled, the CCSN circuit partitions are constructed by tracing the CCC connected to the cell port. This parameter can be used to enable a faster or a more detailed CCSN port-based partitioning algorithm.

This parameter must be used before the char_library command is run.

ccsn_io_mode

`<0 | 1 | 2 | 3 | 4 | 5>`

Enables CCSN model generation when `define_cell -type` is set to UDA or when `char_library -io` is used or the *Inside View* algorithm fails.
Default and recommended: `5`

`0` Disables the IO mode CCSN generation algorithm.

`1-3` These settings are deprecated and are for backward compatibility only.

`4` Enables enhanced recognition of Channel Connected Blocks (CCBs). The sensitization includes:

- Recognition of complementary signals, either external or internally generated.
- Recognition of a larger cone of logic so that all of the CCB side pins are initialized properly for a clean switching CCB output.
- Handles situations where there is no proper inversion in the CCB and inputs to complementary `PMOS` and `NMOS` devices are controlled by interdependent signals that must switch simultaneously for a clean event at the CCB output.
- Handles multiple topologies where transmission gates are involved.
- This setting must be used with `ccsn_allow_multiple_input_switching` set to `2` (or a compatible setting).

`5` Enables enhanced recognition of CCBs as does setting of `4`. In addition, it allows sorting of CCBs depending on the channel path depth (less first), instead of number of inputs (lower number first).

This parameter must be used before the char_library command is run.

ccsn_io_mode_enable

`<0 | 1>` Enables IO CCSN circuit partitioning algorithm.
Default and recommended: `1`

- `0` Do not use the IO CCSN circuit partitioning algorithm when the *Inside View* algorithm is enabled.
- `1` Allow Liberate to use the IO CCSN circuit partitioning algorithm if the *Inside View* algorithm-based partitioning fails to successfully characterize the CCSN.

The generation of CCSN data requires special circuit partitions. When the *Inside View* algorithm of Liberate is enabled, the circuit partitions are constructed from the characterized timing arcs. When the `char_library -io` mode is used, the *Inside View* algorithm is disabled. In this case, the CCSN circuit partitions are constructed from the circuit by tracing the Channel Connected Component/Region (CCC) connected to the cell port.

This parameter must be used before the char_library command is run.

ccsn_load_cap_delta_variation

`<float_number>` Specifies to use 5% of voltage swing as the delta voltage for input and output Piece-Wise Linear (PWL) values.
Default: `0.05`

This parameter must be set before the char_library command is run.

ccsn_load_cap_mode

`<0 | 1>` Controls whether a fresh simulation should be run to determine the CCSN load capacitance values.
Default: `0`

- `0` Enables the backward compatible behavior.

`1`	Specifies to run fresh simulation to determine the CCSN load capacitance values.

This parameter must be used before the char_library command is run.

ccsn_load_cap_slew_width

`<float_number>`	Specifies the width of the slew ramp waveform from initial `0` time point to the end time point while simulating CCSN load capacitance. Default: `10e-12` (that is, `10ps`)

This parameter must be set before the char_library command is run.

ccsn_load_cap_tran_tend

`<float_number>`	Specifies the transient simulation stop time for CCSN load capacitance simulation. By default, the tool considers the estimated stop time for running the simulation, unless it is set to a non-zero value. Default: `0`

This parameter must be set before the char_library command is run.

ccsn_merge_equivalent_stage_tol

`<value>`	Enables merging of equivalent pin-based CCSN stages if they are within the specified tolerance of each other. Default: `0.0`

Setting this parameter to:

- `0.0` disables merging. (Default)
- `0 < value <= 1` specifies a relative tolerance to enable merging.

This parameter must be set before the char_library command is run.

Example

```
# Merge equivalent CCSN stages using a 1% tolerance
set_var ccsn_merge_equivalent_stage_tol 0.01
```

ccsn_miller_init_mode

`<0 | 1 >` Controls whether to use `.nodeset` or `.ic` to initialize the primary input to the Channel Connected Block (CCB) used for measuring the input pin miller capacitance.
Default: `1` (use `.ic`)

- `0` Uses `.nodeset` to initialize the input probe while writing miller decks.
- `1` Uses `.ic` to initialize the input probe while writing miller decks.

This parameter must be set before the char_library command is run.

ccsn_miller_init_vin_thresh

`<value>` Controls how close the inputs are initialized to the starting rail voltage for CCSN miller capacitance characterization. The default value 0.2 initializes input rise to 0.8 Vdd and input fall to 0.2 Vdd.
Default: `0.2`

This parameter must be set before the char_library command is run.

ccsn_miller_slew_width

`<float_number>` Specifies the width of the slew ramp waveform from initial `0` time point to the end time point while simulating CCSN miller capacitance.
Default: `10e-12` (that is, `10ps`)

This parameter must be set before the char_library command is run.

ccsn_miller_vin_mode

`<0 | 1 | 2 | 3>` Controls how input voltage (`Vin`) is measured in the SPICE deck.
Default (and Recommended): `3`

- `0` Specifies that `miller0` uses initial voltage, and `miller1` uses the final voltage (`Vfinal`).
- `1` Specifies that `miller0` uses initial voltage, and `miller1` uses minimum (`Vmin`) or maximum (`Vmax`) voltage depending on the input wire's state (that is, the vector's state). Therefore, for `miller1`, use `Vmax` for falling arcs and `Vmin` for rising arcs.
- `2` Same as `1`, but specifies that `miller1` uses final voltage if CCB is non-inverting. When set, the methodology from `1` is used for negative-unate arcs and the methodology from `0` is used for positive-unate arcs.
- `3` Specifies that `miller0` uses the minimum voltage and `miller1` uses the maximum voltage. When set, `Vin` is acquired using the `deltaV(min,max)` methodology for all arcs.

This parameter must be set before the char_library command is run.

ccsn_miller_vout_delta_variation

`<value>` Controls the output swing for CCSN miller capacitance characterization. For FinFET technologies, the output swing needs to be large enough to model the complete impact of backward miller effect.
Default: `0.5`

Note: The default value `0.5` implies `0.5 * (Vdd-gnd)` output swing.

This parameter must be used before the char_library command is run.

ccsn_model_channel_connected_ccbs

`<0 | 1 | 2>` Controls the modeling of CCBs connected to a channel.
Default: `0`

- `0` Prefer one active CCB closer to the output whenever possible.
- `1` Choose among the two active CCBs based on the worst case noise selection.
- `2` Enable probing on pass gate separated channel connected outputs in the same CCB. This setting creates two active CCBs whenever possible.

This parameter must be set before the char_library command is run.

ccsn_model_passgate_compatibility_mode

`<-1 | 0 | 1>` Controls the modeling of `is_pass_gate` and `has_pass_gate` CSS noise attributes.
Default and recommended: `0`

- `-1` Keep the old behavior for backward compatibility.
- `0` Specify the LC compatible setting where `is_pass_gate` is modeled for single pass gate, and `has_pass_gate` gets applied only when at least one CCB under such pin has `is_pass_gate` set to `TRUE`.
- `1` Include the features specified by value `0`.In addition, provide enhanced handling of the `has_pass_gate` attribute where the `has_pass_gate` attribute of the pin is applied to unbuffered input/output pins.

This parameter must be set before the char_library command is run.

Example

When `ccsn_model_passgate_compatibility_mode` is set to `1`, single passgate `(I --> P --> O)` is modeled as following:

```
pin (I) {
```

```
  has_pass_gate : true
  ...
}
pin (O) {
  has_pass_gate : true
  timing () {
     ccsn_first_stage() {
        is_pass_gate : true
     }
  }
}
```

If `ccsn_model_passgate_compatibility_mode` is set to `0`, `has_pass_gate` under pin `I` will not get applied. Then, passgate is followed by combinational cell `(I -> P -> C -> O)` as following:

```
pin (I) {
  has_pass_gate : true
}
pin (O) {
  timing () {
    ccsn_first_stage () {
        ....
    }
  }
}
```

ccsn_model_related_node_attr

`<0 | 1 | 2>` Controls whether the `related_ccb_node` or `related_spice_node` attribute should be included in the output library for noise models.
Default and recommended: `1`

- `0` Do not include the `related_ccb_node` or `related_spice_node` attribute in the output library for noise models.

1	Include the `related_ccb_node`, `related_spice_node`, `load_cap_rise`, and `load_cap_fall` attributes in the output library. The ccsn_use_io_ccb_format parameter determines which format the CCSN noise model will follow.
2	Include the `related_ccb_node` attribute for both stage-based and referenced CCSN models.

This parameter must be set before the char_library command is run.

ccsn_no_input_dc_current_mode

`<0 | 1 | 2>` Controls the DC current characterization of CCSN for the cells that has no inputs.
Default: `2`

0	Do not perform any simulation reduction and post process for no input cell like tie cell.
1	Post process for tie cell to duplicate the first line across the rest of the DC table.
2	Only simulate first row of `dc_current` table and do post process similar to 1

This parameter must be used before the char_library command is run.

ccsn_one_sided_tristate

`<0 | 1>` Enables checking for one sided tristates.
Default: `0`

In CCSN modeling, the output is needed to be able to swing rail-rail (from a `1` to a `0`) in order to propagate noise pulses. This may not always be possible with tri-state cells. If the output of the CCSN stage can swing to a "Z" state, based on its input switching, then additional side inputs might need simultaneous toggling to pull it to a `1` or a `0`. Setting `ccsn_one_sided_tristate` allows Liberate to evaluate combinations of the primary inputs and tie them together until the Z state is resolved to a `1` or a `0`. This is a pessimistic solution that physically translates to noise affecting multiple inputs at the same time.

This parameter must be used before the char_library command is run.

ccsn_part_mode

`<0 | 1 | 2>` Sets the type of circuit to use for CCSN simulations.
Default: `0`

`0`	Use partitioned circuit ("cut") for CCSN simulations.
`1`	Use original circuit ("uncut") for CCSN simulations.
`2`	Use original circuit ("uncut") for CCSN simulations when CCB has multiple input switching.

This parameter must be used before the char_library command is run.

ccsn_pin_criteria_mode

`<0 | 1 | 2>` Adds additional criteria for modeling CCSN groups as pin-based instead of arc-based.
Default: `2`

`0`	Favor arc-based modeling. Should only be used for backward compatibility purposes to match 12.1.5 or prior releases.
`1`	Performs a set of checks to favor pin-based modeling.
`2`	Performs the checks performed with the setting of `1` and some checks are run on the channel connection and re-convergence. These additional checks are done to determine if a logical condition should be modeled at the arc or pin level.

Multiple criteria may be considered when deciding whether to model a stage as pin-based or arc-based. Setting this parameter adds additional criteria that pushes a stage to the pin.

This parameter must be used before the char_library command is run.

ccsn_pin_high_effort

`<0 | 1>` Selects a CCS effort algorithm.
Default and recommended: `1`

- `0` Restore the behavior prior to 2.4p2 and prior releases.
- `1` Create the CCSN stages input pins that are not modeled by any CCSN stages (either arc based or pin based) The default behavior of Liberate was changed in release 2.5 to create these CCSN stages because this behavior results in a more complete CCSN `.lib` model.

This parameter must be used before the char_library command is run.

ccsn_pin_stage_lshift

`<0 | 1 | 2>` Specifies the methodology for modeling CCSN for level-shifters.
Default (and Recommended): `0`

- `0` Model level-shifters as arc-based or pin-based as per the CCSN specification for that CCB.
- `1` Model level-shifters as pin-based to work around LBDB-716 issues.

The CCSN Characterization guidelines have rules to outline when CCSN stages should be modeled as pin-based and when as arc-based. However, if level-shifters included arc-based CCSN models, older versions of Library Compiler generates "LBDB-716" errors and fail to compile the library. The default setting works around this problem by forcing pin-based models. Library Compiler versions post LIBERATE 12.1 ISR6 does not have this issue,. Therefore, it is recommended to model these as arc-based.

This parameter must be used before the char_library command is run.

ccsn_pin_stage_merge_mode

`<0 | 1 | 2 | 3 | 4>` Controls merging of CCSN data.
Default and recommended: `4`

`0`	Enables behavior prior to release 2.3. (Use only for backward compatibility.)
`1`	Enables behavior of release 2.3. (Use only for backward compatibility.)
`2`	Allows Liberate to also choose inverter outputs whenever possible.
`3`	Enables mode 1 and 2, and for output CCSN stages, chooses inverter input whenever possible.
`4`	Gives preference to probes directly on inverter outputs. This setting provides the expected functionality when the output is also channel connected to an input.

It is often desirable to merge CCSN `stage_type` `pull_up` and `pull_down` data to `stage_type` "both". Use this parameter to instruct Liberate to prefer ccsn_stage_both for input/output probe nodes that use inverter input/outputs.

Set this parameter to enable enhanced conditional checks to understand "don't care" conditions. Previously, Liberate considered two groups to be equivalent (which allowed Liberate to merge them together into one group.) Liberate required all CCR (Channel Connected Regions of transistors) inputs to have the same state even when a particular input is a "don't care" for an output. This caused more CCSN stages.

This parameter must be used before the `char_library` command.

Example

```
set_var ccsn_pin_stage_merge_mode 0
```

ccsn_pin_voltage_level_attrib

`<0 | 1 | 2>` Writes the Liberty attributes `input_signal_level` and `output_signal_level` into CCSN groups.
Default and recommended: `1`

- `0` Do not output `input_signal_level` and `output_signal_level` attributes into the library.
- `1` Output `input_signal_level` and `output_signal_level` pin attributes into the CCSN (`ccsn_first_stage` and `ccsn_last_stage`) groups. This is compatible with the 2013.03 version of LC.
- `2` Update all CCSN stages (arc and pin) with the CCSN signal attributes regardless of the multi-domain status of the cell.

This parameter must be used before the `char_library` command.

ccsn_prefer_min_vt_probe

`<0 | 1>` Methodology for determining CCB output probe point selection.
Default: `0`
Recommended: `1`

This parameter is used when a CCSN stage (Channel Connected Region) has more than one output that can be selected as the output probe.

- `0` (Worst-case noise model) Select the output probe node that results in the largest propagated noise, even if a full voltage swing is not possible such as when the output is in the middle of an `nfet` or `pfet` stack.

1	(Noise-on-delay model) Liberate selects the output probe for the CCSN stage that connects to both NMOS and PMOS devices whenever possible. This node may not produce the largest propagated noise but it maximizes the observed output voltage swing. This might mean that the probe point is not the one that is most susceptible to noise but may have better results in noise-on-delay analysis.

This parameter must be used before the char_library command.

ccsn_prefer_pin_termination

<0 | 1>

Chooses the worst-case CCSN stages under a pin based on the probe node. Sometimes, there are differences in Vout and prop template sizes when choosing from single-stage and multi-stage CCBs.
Default: 0

0	Choose the worst-case CCSN stage under a pin based on the probe node.
1	When choosing from the CCSN stages with the same condition, priority is given to the one that ends on a cell pin/port when one is available. In rare cases, the worst-case CCSN group under the pin switches between a single-stage and a multi-stage CCSN entry, which can lead to a difference in the number of Vout groups. The setting of 1 will choose a stage without considering its voltage rise/fall values, which could cause optimism. Use this setting only after analyzing the CCSN and confirming that the new results are acceptable.

This parameter must be used before the first write_library command is run.

ccsn_print_is_needed_if_false_attr_value

`<0 | 1>` Controls the output of the `is_needed` attribute in CCSN groups in the output library when this attribute has a value of `false`. This indicates that the CCSN group is not needed.
Default: `0`

- `0` Do not model ccsn groups when the Liberty attribute `is_needed` has a value of `false`.
- `1` Write ccsn_first_stage/ccsn_last_stage groups into the output library when the Liberty attribute `is_needed` has a value of `false`. This setting may clear up library compilation issues.

This parameter affects cells with pins that do not require CCSN data. Print the following ccsn group with `is_needed` pin attribute.

```
pin (a) {
  ccsn_first_stage () {
    is_needed :  false;
  }
}
```

Note: The `ccsn_*` groups with the attribute `is_needed` with value `false` cannot contain miller cap or any such attribute ("when" may be allowed).

This parameter must be used before the char_library command is run.

ccsn_probe_enable_toggle_res_check

`<0 | 1>` Specifies whether to enable or disable the checks for wire toggle resistance. If enabled, the checks are performed while selecting the probe nodes especially when there are multiple switching output wires.
Default (and Recommended): `1`

- `0` Disables the checks for wire toggle resistance to maintain consistency across the PVTs because different PVTs (even only voltage changes) can result into different wire toggle resistances.

`1`	Enables the checks for wire toggle resistance. This setting is recommended because it often helps to identify the worst case.

This parameter must be used before the char_library command is run.

ccsn_probe_non_gate

`<0 | 1>` Controls whether gate and channel connections should be probed for activity when the pass gate is non-conducting. Default (and Recommended): `1`

On the CCSN side, there are designs where the output of say a CCB on D directly goes into a pass gate channel with the other side channel connecting to a gate. Liberate by default only considers gate connections to probe for activity as a simulation wire. This misses all CCB activities on pin D when the pass gate is non-conducting. The `ccsn_probe_non_gate` parameter allows Liberate to monitor the other end of the pass gate as well.

`0`	Do not generate CCSN when no probe can be identified at the input to a gate.
`1`	Permits probe selection of non-gate connected wires. This will generate CCSN stages on inputs such as the pin D of flip flops. This setting also helps clear the Library Compiler LBDB-712 messages that might lead to false failures because the output probe is always disconnected from any gate and the glitch (as characterized with the specific boundary conditions) cannot cause any state change internally. The glitch analysis tool might not be aware of this.

Liberate models CCSN `ccsn_*_stages` only when the noise can reach a gate (or primary output). For certain inputs, such as the D of a flip-flop, there might be no characterized arcs or vectors. When the older CCSN format (`ccsn_use_io_ccb_format=0`) is chosen, in the above case, the CCSN IO mode algorithm is triggered for D, resulting in probe of the input for Q. The same approach works with the new CCSN format

(`ccsn_use_io_ccb_format=1`) because the receiver capacitance and timing arcs need to share vectors with the CCSN stages.

This parameter must be used before the char_library command is run.

ccsn_prop_noise_check_criteria_mode

`<0 | 1>` Controls whether the enhanced check and retry method should be enabled for CCSN propagated noise characterization.
Default: `0`

`0`	Uses the default method.
`1`	Enables the check of propagated noise waveform and ensures that the peak value is not greater than 0.8*VDD and not less than 0.05*VDD.

Note: This parameter is supported only in Liberate and Liberate MX.

This parameter must be used before the char_library command in Liberate, or the `char_macro` command in Liberate MX.

ccsn_prop_noise_peak_mode

`<0 | 1>` Used with CCSN characterization of level shifters if Library Compiler issues warnings about incorrect `index_1` voltage ranges.
Default: `0`

`0`	Used with CCSN characterization of level shifters.
`1`	Control the propagated noise input values

This parameter must be used before the char_library command is run.

ccsn_prop_retry_duration_incr

`<0 | 1 | 2>` Automatically extends the simulation duration for CCSN measurements if the original simulation fails. If you see warnings like "*Unable to obtain reasonable CCS noise propagated waveform*", set this parameter and re-run characterization.
Default: `1`

- `0` Disables the algorithm and does not automatically extend the simulation duration.
- `1` Some CCSN stages at special Vt might produce non-triangular outputs with some amount of distortions and have multiple cliffs or direction changes. These stages may need longer simulation time. Use this setting to detect such situations and automatically extend the simulation duration.
- `2` This setting is for backward compatibility with the LIBERATE 18.1 ISR2 and prior releases where the check is designed for well-behaved responses and you can choose to not increase the tend times as needed.

This parameter must be used before the char_library command is run.

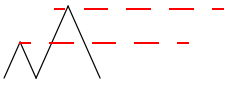
ccsn_prop_retry_peak_incr

`<value>` Controls the input PWL generation for propagated noise. This is the percentage of VDD-VSS voltage swing (for that partition) used to increment the peak during the next retry attempt. A larger value results in larger peaks being generated for the input, when the initial set of inputs fails to produce the desired output signals.

Example:

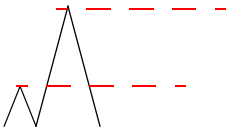
Given vdd=1, gnd=0, and initial peak=0.1

If ccsn_prop_retry_peak_incr = 0.2



... and the second would be 0.1 + (1 * 0.2) = **0.3v**
The first peak would be at 0.1

If ccsn_prop_retry_peak_incr = 0.8



... and the second would be 0.1 + (1 * 0.8) = **0.9v**
The first peak would be at 0.1

This parameter must be used before the char_library command is run.

ccsn_prop_tristate_meas_mode

`<0 | 1>` Controls whether propagated noise groups on single-sided CCBs should be shaped.
Default: `0`

`0`	Do not help shape propagated noise groups.
`1`	Helps shape propagated noise groups on single sided CCBs.

This parameter must be used before the char_library command is run.

ccsn_prune_last_stage

`<0 | 1>` Controls skipping of the *ccsn_last_stage* generation on output pins.
Default: `1`

- `0` Use this setting for backward compatibility with release 2.3 and prior behavior where the last stage is not skipped.
- `1` Liberate skips *ccsn_last_stage* generation on some output pins. When enabled, causes Liberate to skip generation of all pin-based CCSN stages for output pins that are not destinations of timing() arcs and outputs two sided (stage_type: both) CCSN stages, whenever possible.

This parameter must be set before the char_library command is run.

ccsn_related_power_pin_check

`<0 | 1>` Controls whether the `related_power_rails` used by all the pins across the cell should be checked to determine the input/output signal level used in the CCSN stage/input/output CCB.
Default: `1`

- `0` Use this setting for backward compatibility with the LIBERATE 18.1 ISR1 and prior releases where no check is performed.
- `1` Check the `related_power_rails` used by all the pins across the cell to determine the input/output signal level used in the CCSN stage/input/output CCB. The `signal_level` attribute is written out only if more than one power rail is used in the cell.

This parameter must be set before the char_library command is run.

ccsn_simultaneous_switch_save_vecdata

`<0 | 1>` Enables the generation and saving of `vecdata` for select pins. The `vecdata` is required when generating CCSN data for an arc in a library.
Default and recommended: `1`

- `0` CSN might not be generated for the `select` pin of a MUX when `simultaneous_switch_from_cell_when=1`. This setting is provided for backward compatibility to LIBERATE 12.1 ISR4 and prior releases.
- `1` Generate and save all `vecdata` for `select` pin even when `simultaneous_switch_from_cell_when=1`.

This parameter must be used before the `char_library` command.

ccsn_slew_based_output_voltage

`<0 | 1 | 2>` Specifies the modified thresholds to use for output_voltage in input/output_ccb CCS-N models.

The ccsn_first_stage/ccsn_last_stage recommends 10/30/50/70/90 measurement thresholds, regardless of library thresholds. The input_ccb/output_ccb CCSN specification allows for the thresholds to match library thresholds.
Default: `2`

`0` Use 10/30/50/70/90 thresholds, regardless of CCSN format or slew thresholds.

`1` Use 10/30/50/70/90 thresholds if model format is the ccsn_*_stage format or if measure slew thresholds are 30/70; otherwise use the following:

- Rise: `(slew_lower_threshold/2, slew_lower_threshold, 0.50, slew_upper_threshold, (1 + slew_upper_threshold)/2 ) of VDD`
- Fall: `((1 + slew_upper_threshold)/2, slew_upper_threshold, 0.50, slew_lower_threshold, slew_lower_threshold/2 ) of VDD`

`2` Similar to `1`, but use `output_threshold_pct_rise` and `output_threshold_pct_fall` instead of `0.5*VDD`.

This parameter must be used before the `char_library` command.

ccsn_slew_load_mode

`<0 | 1 | 2 | 3>` Determines output voltage slew and load
Default: `0`

Note: This parameter is set to `3` automatically if you are using IO / CCB format or are characterizing both the formats.

`0` 2x2 vectors.

Use 0.05*max_pin_transition and 0.1*max_pin_transition for input slew.

Use 0.05*max_pin_cap and 0.1*max_pin_cap for load_cap if CCB output is an output pin.

Use zero value for load capacitance if CCB output is an internal net.

`1` 2x2 vectors.

Use the second and middle index points of CCS timing table for input slew.

Use the second and middle index points of CCS timing table for load capacitance if CCB output is an output pin.

Use zero value for load capacitance if CCB output is an internal net.

`2` nxn vectors.

Indexes of output voltage exactly match the timing arc indexes. Each (slew, load) pair in the timing table has a corresponding output voltage vector.

Note: `ccsn_vout_slew_size` and `ccsn_vout_load_size` not supported in this mode.

`3`	Support user-defined vector size. The number of slew and load is determined by `ccsn_vout_slew_size` and `ccsn_vout_load_size`. However, the range is from `2` to the number of timing arc indexes. If the user-defined size is less than `2`, use `2`. If the user-defined size is larger than the number of timing arc indexes, use the number of timing arc indexes. Slew and loads are selected from timing arc. Use a zero for load capacitance if CCB output is an internal net.

This parameter must be used before the `char_library` command.

ccsn_sort_merge_hidden_mode

`<0 | 1 | 2>` Controls pin-based CCSN structural compatibility of CCSN models that are generated.
Default and recommended: `0`

`0`	Sorts the arcs to ensure consistency in CCSN groups in LIBERATE 16.1 and later releases.
`1`	Maintains compatibility with the 3.2p4_lcs release.
`2`	Maintains compatibility with the LIBERATE 12.1 ISR2 to LIBERATE 15.1 ISR5 releases.

This parameter must be used before the `char_library` command.

ccsn_switch_cell_partition_mode

`<0 | 1>` Enables better handling of power gates for DC current characterization.
Default: `0`

- `0` Does not distinguish between power gating MOS and switched rail powered devices for cell level `dc_current` generation purposes. Iterates across all possible gate connections to obtain the `dc_current`.
- `1` Chooses the gate terminal of the power gating MOS device for the `dc_current` characterization. It handles the inverter chain appropriately for first and last stage CCSN characterization on the respective input and output pins. You can use enhanced identification of first and last stage Channel Connected Blocks (CCBs) on switch rail powered gates, and thus avoid treating them as a single simple CCB.

This parameter must be set prior to the `char_library` command.

ccsn_uda_user_probe_mode

`<0 | 1>` Instructs Liberate to use the physical node from the internal wire specified as the CCSN probe node in the define_arc `-ccsn_probe` command. For `ccsn_first_stage`, this is the output from the first stage. For `ccsn_last_stage`, this is the input to the last stage.
Default: `0`

- `0` Liberate follows the algorithm specified by the ccsn_probe_mode parameter, and by default, ignores the `-ccsn_probe` option of the `define_arc -type ccsn` command.
- `1` Liberate attempts to honor the internal wire from the `define_arc -ccsn_probe` command. For this setting, ensure that the `-user_arcs_only` option is specified with the `char_library` command.

This parameter must be set before the char_library command is run.

ccsn_use_io_ccb_format

`<0 | 1>` Specifies CCS Noise characterization and modeling methodology.
Default: `0` (Use stage-based format)

Recently, a new format for modeling CCS Noise was introduced to replace the `ccsn_first_stage` and `ccsn_last_stage` formats with the `input_ccb` and `output_ccb` formats. A number of methodology changes for how CCSN should be characterized were also implemented. This parameter specifies which format to use for characterization and modeling. As per the specification, the formats are incompatible, and so the decision must be made prior to characterization.

- `0` Characterize and model as per the stage-based format.
- `1` Characterize and model as per the CCB-based format.

The CCB-based format offers a number of improvements over the stage-based format, some of which were only previously available to customers using CDB/ECSM-N. However, since adopting this new format means updating most of the tool-chain, Cadence recommends that customers allocate sufficient time and resources for the necessary qualifications prior to making any decision on production use.

This parameter must be set before the char_library command is run.

ccsn_use_output_voltage_level

`<0 | 1 | 2>` Controls the voltage level format to output in the liberty file generate after the characterization.
Default (and Recommended): `1`

- `0` Liberate characterizes CCB transient data using the minimum of the associated voltage domains. This is done to ensure compatibility with the checks being done in Synopsys Library Compiler (LC).
- `1` If present, apply the `input_signal_level` and `output_signal_level` (see ccsn_pin_voltage_level_attrib) to the CCB that is now supported by LC.
- `2` Check the voltage swing observed on the CCB output. When the voltage swing does not match the expected `input/output_voltage_level`, fall back to using the smaller among the `input/output_voltage_level` and available supplies. This might be acceptable for older style CCSN libraries (with a potential loss of quality), but is not compatible with the newer CCSN IO CCB model format (see ccsn_use_io_ccb_format) due to the newer standards/guidelines.

This parameter must be set before the char_library command is run.

ccsn_vector_define_mode

`<0 | 1>` Determines whether the `-when` or the `-vector` option specified with the `define_arc` command will be honored.
Default: `0`

The CCSN group generation is based on arcs scheduled for characterization. The vectors can be specified directly using the `define_arc -type ccsn` command or by the related delay and power `define_arc` commands. Sometimes, the CCSN stimuli are not generated as expected when the `define_arc` commands have a `-vector` option and no `-when` option. Use the `define_arc` command with the `-type ccsn` option to tune the CCSN vector selection.

- `0` The CCSN vector selection algorithm observes the timing as per the condition defined with the `define_arc -when` command. The values in the `define_arc -vector` for the side pins are ignored, only the `-when` option will be honored.
- `1` The CCSN vector selection algorithm observes the values in the `define_arc -vector` for the side pins. The values in the `-when` options might not be applied. This setting can lead to more and/or different CCSN groups being identified for arcs that are specified with the `-vector` option of a `define_arc` command. The `-vector` option will be honored.

This parameter must be set before the char_library command is run.

ccsn_vout_load_size

<value> Sets the CCSN output load capacitance size.
Default and recommended: `0`

When this parameter is set to value `n`, the size is applied when ccsn_slew_load_mode=`3`.

ccsn_vout_output_precision

<value>	Specifies the upper limit of the number of digits of precision for CCSN vout data. Default and recommended: 3

This parameter controls the upper limit for the number of printed digits while generating the newer CCSN model (see ccsn_use_io_ccb_format). When you set this parameter to a value lesser than the default value, no advantage is gained because the default meets the Liberty requirements. Increasing this might not add any appreciable increase to accuracy, but might increase the chances of Liberty incompatibility instead when more digits are printed.

Example

```
set_var ccsn_vout_output_precision 3
```

ccsn_vout_slew_size

<value>	Sets the CCSN input slew size. Default and recommended: 0

When this parameter is set to value n, the CCSN input slew size is applied when ccsn_slew_load_mode=3.

ccsn_xfr_ccc_probe_mode

<0 | 1>	Ensures that, when modeling CCSN first-stage groups, Liberate does not probe transmission gate outputs when the transmission gate input was the arc pin. Default: 1

0	Restores the behavior of the 3.0 and prior releases.
1	Disables probing of transmission gate outputs when modeling CCSN first-stage groups

This parameter must be set before the char_library command is run.

ccsp_base_curve_points

<value>	The maximum number of points in a CCSP base curve. Default: `15`

When generating CCSP curve data, Liberate can search for common curve shapes. When found, the common shape can be stored into a lookup table and reused multiple times. This is done to reduce the size of the Liberty file. Liberate uses up to the specified number of points in a shared base curve.

This parameter must be set before the write_library command is run.

ccsp_default_group

`<min | max | off>`	Determines whether and how to write a default CCSP power group into the output library (see write_library). Default: `off`

`min`	Selects the `dynamic_current` that has the minimum `internal_power` to write into a default CCSP power group in the output library.
`max`	Selects the `dynamic_current` that has the maximum `internal_power` to write into a default CCSP power group in the output library.
`off`	Disables the feature of writing a default CCSP power group into the output library.

This parameter must be set before the write_library command is run.

Example

```
# Enable a max default ccsp power group
set_var ccsp_default_group max
```

ccsp_intrinsic_res_criteria

`<off | min | max | avg>`

Determines which CCSP intrinsic resistance values should be selected for writing out to the library if there are multiple supplies with the same output.
Default: `min`

`off`	Writes out all intrinsic resistance groups, that is, the original data.
`min`	Writes out only the minimum values.
`max`	Writes out only the maximum values.
`off`	Writes out only the average values.

This parameter must be set before the write_library command is run.

Example

Example 1

When `ccsp_intrinsic_res_criteria=off`, all intrinsic resistance groups are written out:

```
intrinsic_parasitic () {
  when : "(c * sn)";
  intrinsic_resistance (VDD) {
    related_output : q0;
    value : 0.443134;
  }
  intrinsic_resistance (VDD) {
    related_output : q1;
    value : 0.443134;
  }
  intrinsic_resistance (VSS) {
    related_output : q0;
    value : 0.702104;
  }
  intrinsic_resistance (VSS) {
    related_output : q1;
    value : 0.702104;
```

```
    }
  }
```

When `ccsp_intrinsic_res_criteria=min`, only the minimum intrinsic resistance groups are written out:

```
  intrinsic_parasitic () {
    when : "(c * sn)";
    intrinsic_resistance (VDD) {
      related_output : q0;
      value : 0.443134;
    }
    intrinsic_resistance (VDD) {
      related_output : q1;
      value : 0.443134;
    }
  }
```

When `ccsp_intrinsic_res_criteria=max`, only the maximum intrinsic resistance groups are written out:

```
intrinsic_parasitic () {
    when : "(c * sn)";
   intrinsic_resistance (VSS) {
      related_output : q0;
      value : 0.702104;
    }
    intrinsic_resistance (VSS) {
      related_output : q1;
      value : 0.702104;
    }
  }
```

When `ccsp_intrinsic_res_criteria=avg`, only the average intrinsic resistance groups are written out:

```
intrinsic_parasitic () {
    when : "(c * sn)";
    intrinsic_resistance (VDD) {
      related_output : q0;
      value : 0.572619;
    }
    intrinsic_resistance (VDD) {
      related_output : q1;
      value : 0.572619;
```

```
        }
    }
```

ccsp_leakage_current_abstol

<value>	Sets an absolute tolerance of when Liberate adjusts CCSP leakage current for Library Compiler compliance. The value `1.0e-10` conforms to the requirements in Library Compiler 2008-09 SP1. The value `0` conforms to the older releases of Library Compiler such as 2007-12 SP1. Default: `0`

This parameter must be used before the `char_library` command.

ccsp_leakage_current_compensation_mode

`<0 | 1 | 2>` Controls the methodology used for compensating `leakage_current` values for Library Compiler compliance. Default and recommended: `1`

- `0` — Adjust the rail with smallest `leakage_current`.
- `1` — Adjust a 0v rail. If multiple 0v rails are attached to a cell, then adjust the primary_ground with the *most* `leakage_current`.

  Note: Some adjustments may be the result of truncation errors due to unit scale. For 28nm libraries and smaller geometries, we recommend using the following units prior to changing this parameter:

  ```
  set_units -timing 1ps -capacitance 1ff
  -current 1ua -leakage_power 1pw
  ```

  This `set_units` change usually has the added benefit of decreased file size with the same (or slightly improved) accuracy.
- `2` — Reduces the largest magnitude leakage current values enough to produce a zero sum, but not farther than 0. For header and footer cells, it removes the virtual rail current from the largest supply, and and also from the gate leakage, if necessary. In all cases, retains the correct sign on all currents.

This parameter must be set before the write_library command is run.

ccsp_meas_supply_cap_sim_duration

<value> — Defines the duration of the supply capacitance measurement simulation.
Default: `1e-11` in seconds

You can characterize the intrinsic capacitance of a supply pin (see set_pin_capacitance `-meas_supply_cap`). To specify the time in seconds that this supply pin should take to be ramped, you can use the `ccsp_meas_supply_cap_sim_duration` parameter.

See the ccsp_meas_supply_cap_ramp_ratio parameter to specify the ratio of the supply for the ramp.

This parameter must be used before the char_library command is run.

ccsp_meas_supply_cap_ramp_ratio

<value> — Defines the supply voltage ramp percentage. If VDD, ramp to `0.9*VDD`; if VSS, ramp it to `0.1(VDD-VSS)`.
Default: `0.1`

You can characterize the intrinsic capacitance of a supply pin (see set_pin_capacitance `-meas_supply_cap`). To specify the voltage swing of the supply ramp as a ratio of the supply, you can use the `ccsp_meas_supply_cap_ramp_ratio` parameter.

See the ccsp_meas_supply_cap_sim_duration parameter to specify the duration of the supply capacitance measurement simulation.

This parameter must be used before the char_library command is run.

ccsp_min_pts

<value> — Sets the minimum number of desired CCSP samples. This parameter only has an effect when the `ccsp_segmentation_effort` parameter is enabled.
Default: `6`

This parameter must be used before the `char_library` command.

ccsp_pin_direction_post_default

`<0 | 1>` Controls finding unateness of CCSP dynamic groups.
Default: `0`

`1` When determining the pin direction (rise or fall) of the CCSP power group, Liberate examines the default timing group when no state-dependent timing group matches the "when" condition in the ccsp power group.

This parameter may be used before the char_library command.

ccsp_prune_factor

<value> Set the CCSP tail decay factor.
Default: `2`

Use this parameter to control the decay to be applied after the `ccsp_prune_second_tol`. If this parameter is set to `2`, the ccsp waveform is pruned at `ccsp_prune_factor*(ccsp_prune_start_tol - ccsp_prune_second_tol)` after `ccsp_prune_second_tol`. See ccsp_prune_start_tol.

This parameter must be used before the `char_library` command.

ccsp_prune_second_tol

<ratio> Set the second CCSP tail prune tolerance.
Default: `0.03 (3%)`

This parameter must be used before the `char_library` command.

ccsp_prune_start_tol

<ratio> Set the first CCSP tail prune tolerance.
Default: `0.0 (0%. Pruning disabled)`

If set to a value greater than `0`, Liberate terminates the CCSP waveform early. Liberate follows the raw current waveform backwards from the end to find the point where the

waveform is the `ccsp_prune_start_tol` of the waveform peak. Liberate then follows the current forwards to the point where it reaches `ccsp_prune_second_tol` (default `3%`) of the peak. The difference between these two points is the elapsed time (`"x"`). Assuming exponential decay, Liberate terminates the waveform after a time `2x` (see ccsp_prune_factor) from the `ccsp_prune_second_tol` point.

When this parameter is set to `0` (default), no waveform trimming is done.

Note: `ccsp_prune_second_tol` must be less than `ccsp_prune_start_tol`. If this check fails, Liberate automatically adjusts `ccsp_prune_second_tol` to `0.5 * ccsp_prune_start_tol`.

Setting this parameter to the desired value (that is `0.10`, for `10%` of peak) would act as described above to prune the tail of the CCSP waveform. The pruning is performed on the raw current waveforms and is done before any other CCSP waveform shaping methods. This waveform pruning has been useful to trim long waveform tails that have caused problems in some power tools.

Liberate also checks for the starting leakage value in calculating the percentages. For example, 10% prune time is reached at time `t` when the `i(t) - i(0) < 0.1 *( peak - i(0))`.

This parameter must be used before the char_library command is run.

ccsp_quantization_num_steps

<value>	Set the number of CCSP quantization steps used to preprocess the supply current waveform to smooth out kinks. Default: `100`

This parameter must be used before the `char_library` command is run.

ccsp_rel_tol

<ratio>	Set the CCSP relative tolerance of the area segmentation current as compared against the original area of the supply current. Default: `0.02`

This parameter must be used before the char_library command is run.

ccsp_related_pin_mode

`<0 | 1 | 2 | 3>` Controls modeling of dynamic current groups for CCSP. In CCSP, dynamic_current groups contain the power data for switching events.
Default: `1`
Recommended: `2`

- `0` Writes out separate `dynamic_current` groups for each output pin. Backward compatible to LIBERATE 14.1 ISR2 and prior releases. (**Deprecated. Should be used only for regression purposes.**)
- `1` Enables combining the dynamic current groups under each output pin into one common group whenever possible. (Deprecated. Should be used only for regression purposes.)
- `2` Combines `dynamic_current` groups into a common group where possible. If multiple input pins switch as in the case of CLK/CLKB or a one-hot MUX, then only the group on the first alphabetical `related_pin` is modeled.

  When there are multiple inputs switching, check if there is a relevant internal_power group (NLPM) defined in the library. If there is, then use the default behavior.

3	Similar to 2, but the `related_inputs` and `input_switching_condition` attributes of the `dynamic_current` group accurately reflects all switching activity. This is the most accurate CCSP format, but is currently not supported in all downstream tools.

Before setting the `ccsp_related_pin_mode` parameter to 3, verify with the tool vendor whether the CCSP format that will be enabled is supported.

Liberate writes out separate dynamic current groups for each output pin. This can give wrong power reports when using CCSP format data.

This parameter must be set before the write_library command is run.

ccsp_segmentation_effort

`<0 | 1 | 2 | 3>` Enables a more accurate CCSP segmentation algorithm. Default and recommended: `3`

0	Disables segmentation for supply current waveforms.
1	Liberate segments the supply current waveform and minimizes the number of points
2	This is currently just for the engineering use. The recommendation is not to use this setting.
3	Uses more points when segmenting supply current waveforms. Increasing the number of points provides the best balance of improved accuracy and file size, with no discernible impact on runtime

This parameter must be used before the char_library command.

ccsp_table_reduction

<0 | 1 | 2>	Enables reduced CCSP table data. Default: 1
0	Disables table size reduction.
1	Use a 3x3 table for CCSP data.
2	Enables a 2x2 table

The Liberty specification for CCSP allows for the table size to be reduced. Power results are usually not significantly impacted by using a smaller data table size. A smaller table size will result in a smaller file size.

This parameter must be used before the char_library command.

ccsp_tail_tol

<*value*>	Set the percentage of area considered as the waveform "tail". The tails are discarded during the preprocessing of CCSP waveforms. Default: 0.05

This parameter must be used before the char_library command is run.

ccsp_unateness_infer_mode

`<0 | 1>` Enables better understanding of the power group unateness. Default: `0`

Unlike timing groups, power groups do not have a direct unateness attribute. Therefore, the postprocessing procedure needs to run extra steps to understand input/output directional relationships are run during post-processing.

`0` Uses a simplified algorithm to infer the unateness from power groups.

`1` Enables running of extra code to infer the unateness of non-unate power groups. This setting has no impact other than better understanding of the directional relationships.

Note: When ccsp_related_pin_mode is greater than `1`, the setting of `ccsp_unateness_infer_mode` is overridden to `1`.

This parameter must be used before the char_library command is run.

cell_port_case

`<upper | lower | preserve>`

Provides control for how all cell pins are output.
Default: `preserve`

`upper`	Convert cell pin names to all upper case in the output ldb and library.
`lower`	Convert cell pin names to all lower case in the output ldb and library.
`preserve`	Keeps the current behavior of maintaining the case from the input netlist.

Important

Do not use `cell_port_case` with Spectre. Spectre is case-sensitive and could have multiple port names that differ only in case.

This parameter must be used before the `read_spice` command.

char_mos_term_cap

`<0 | 1 | 2>`

Enables different algorithms for estimating the transistor device pin capacitance. The capacitance is used at various places in the Liberate algorithm to provide equivalent capacitance when the effective loading of the capacitance is required.
Default and recommended: `2`

`0`	Enables the biased method.
`1`	Restores the behavior of the LIBERATE 13.1 ISR3 and prior releases.
`2`	Enables the integration method, which is used automatically if `extsim_exclusive=1`.

This parameter must be used before the `char_library` command.

char_mos_term_cap_skip_names

`<list>` — Specifies a list of leaf cells to skip for pre-characterization. (see char_mos_term_cap)
Default: `{}` (do not skip any)

Liberate applies a pre-characterization step where individual transistors are simulated to determine basic electrical qualities like pin capacitance. Sometimes, the pre-characterization step might fail and Liberate will generate a warning message such as the one illustrated below. In case this case, set the `char_mos_term_cap_skip_names` parameter.

Example message:

```
WARNING (LIB-0423) char_library: The MOS terminal capacitance pre-characterization
failed for the models or leafcells listed below. This may not be needed to produce
a good library. Add these to the 'char_mos_term_cap_skip_names' parameter to skip
these failed simulations in the next run.
    : nmos_a pmos_a
```

If the above message appears, then add the following command to your Tcl script and rerun the pre-characterization step:

```
set_var char_mos_term_cap_skip_names "nmos_a pmos_a"
```

By doing so, a default MOS terminal capacitance is used for the *Inside View* algorithm.

This parameter must be used before the `char_library` command.

cleanup_tmpdir

`<0 | 1>` — Allows Liberate to delete temporary data directories.

When using external simulators, Liberate creates temporary directories and store data files in them (example: `altos.<pid>.<index>`)
Default: `1`

- `0` — The temporary directories are not removed.

  Note: The temporary directories are not removed when Liberate exits due to a simulation error or when terminated by the host system
- `1` — Liberate attempts to remove temporary directories when it exits

This parameter must be used before the `char_library` command.

client_pending_timeouts

`<value>` Specifies the time after which a pending job process will be stopped. Default: `3600`

Note: The timeout setting does not apply to non-bsub (LSF) systems.

combinational_out_to_out_arc

`<all | all_timing | buf | none>`

Controls the stage-types that the *Inside View* algorithm considers when determining output-to-output arcs. Default: `all_timing`

`all`	All feasible output to output arcs are generated, regardless of circuit topology.
`all_timing`	Similar to `all`, but only timing arcs are generated. This avoids double-counting power arcs.
`buf`	Limits checking for output-to-output arc to single input gates between the two outputs, such as a buffer or inverter.
`none`	Output-to-output arc is disabled, except for user-defined ones using `define_arc`.

This parameter must be used before the `char_library` command.

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` parameter to specify how to model these combinational arcs.
Default: `1` (enabled)

- `0` — Instructs the *Inside View* algorithm to characterize only the active arcs on asynchronous pins in sequential cells.
- `1` — Characterizes the asynchronous arcs for the active edge of the asynchronous pins and the combinational arcs on sequential cells.

Note: If combinational arcs for asynchronous pins are specified using the `define_arc` command, these arcs are characterized without considering the value of this parameter.

This parameter must be specified before the `char_library` command.

Example

```
#Disable combinational_rise or combinational_fall arcs for preset and clear pins
set_var combinational_risefall 0
```

conditional_arc

<0 | 1>

Control conditional arcs. All states are characterized using a proprietary algorithm which results in a faster runtime but only one state being stored in the ldb. This reduces the size of the ldb and also results in all output libraries being generated with no state dependency.
Default: 1 (output conditional arcs)

- 0 — Turn off the storing of conditional arcs in the library database. When set to 0, this parameter overrides conditional_constraint.
- 1 — Outputs conditional arcs with full state dependency.

This parameter must be used before the char_library command.

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 parameter 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 parameter must be specified before the char_library command.

conditional_cap_hidden_pin_mode

`<0 | 1 | 2 | 3>` Keeps or removes the output pin(s) from the `when` of the receiver capacitance groups in the output library.
Default: `0`

`0`	Removes the output pin from the `when` of the CCS and ECSM receiver capacitance.
`1`	Removes the output pin from the `when` of the CCS and ECSM receiver capacitance with better structural matching across PVT. **Note:** This is an improvement over the setting of `0`.
`2`	Allows the output pins in the `when` of the CCS and ECSM receiver capacitance.
`3`	Allows the output pins in when condition with better structural matching across PVT. **Note:** This is an improvement over the setting of `2`.

When a setting of `0` or `1` is used to remove the output pins, the parameter `conditional_rcvr_cap_select_criteria` can be used to select one of the overlapping capacitance groups to model in the output library.

Note: The settings of `0` and `1` can create libraries with receiver capacitance groups that have the same `when` condition. For example, if the hidden power states `D&Q` and `D&!Q` are characterized and the output pin is removed, there are two receiver capacitance groups both with a when of `D`. These two groups may have different pin capacitance values. It is up to the timer to choose which one to use.

This parameter must be specified after the `char_library` command.

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 parameter must be set before the `write_library` command.

conditional_constraint

`<0 | 1 | 2>` Controls conditional constraint characterization for timing constraints.
Default: `1 (on)`

`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 parameter is used to enable characterization of conditional (state-dependent) timing constraints including `setup`, `hold`, `recovery`, `removal`, and `minimum pulse width`.

This parameter must be specified before the `char_library` command.

Example

```
# Enable conditional arcs for timing constraints
set_var conditional_constraint 1
```

conditional_expression

`<merge | merge_min_max | separate | single>`

Controls the set of `when` conditions or states that should be characterized and modeled separately for an arc.
Default: `merge`

`merge` Allows a `when` to have an OR'd list of states, where those states are equivalent in terms of their internal switching nodes and loads.

`merge_min_max`

Ensures that if the number of arcs with different `when` conditions/states from `merge` exceed the value set by the `conditional_expression_max_whens` parameter, then the number of state-dependent arcs is reduced to a similar number of minimum and maximum states such that the total number does not exceed `conditional_expression_max_whens`. An estimated delay is used to determine these states. This avoids excessive simulation and modeling time while maintaining the `min` and `max` timing groups so that `min_max` default tables can be produced (see set_default_group).

`separate` Ensures that for each arc with a `when`, the `when` condition is a single state (an AND'd list of pins). The `when` condition will not contain multiple states OR'd together. Multiple vectors can be associated with a `when` if they differ in internal pin settings not represented in the `when` condition.

`single` Forces to use only a single `when` condition.

This parameter must be used before the `char_library` command.

Example

```
# Split groups with OR'd conditions
set_var conditional_expression separate
```

conditional_expression_max_whens

`<value>` Specifies the maximum number of timing groups to model with unique `when` conditions.
Default: `256`

Note: The specified value should be greater than or equal to `1`.

When you specify the maximum number of state-dependent timing groups that can be modeled, Liberate determines all of the unique `when` conditions. If the number of these unique conditions exceeds the limit set by `conditional_expression_max_whens`, a warning is issued. In addition, the number of `when` conditions to model is automatically reduced based on the `conditional_expression` setting such that `separate` is tried first and then `merge` is used.

This parameter must be set prior to the `char_library` command.

conditional_hidden_power

`<0 | 1>` Controls state dependence of hidden power.
Default: `1` (output state-dependent hidden power)

- `0` Characterize and model only unconditional (state-independent) hidden power. If the define_arc command is used to specify hidden power arcs, its `-when` option will be ignored when determining what states to simulate. To control the simulated vectors used to characterize hidden power, use the `-logic_condition` and `-vector` options of the `define_arc` command.
- `1` Enable characterization and modeling of state-dependent hidden power.

This parameter must be used before the `char_library` command.

Example

```
# Turn off conditional states for hidden power
set_var conditional_hidden_power 0
```

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.

This parameter must be used before the `char_library` command.

Example:

```
# Disable constants in 'when' conditions
set_var conditional_include_constant 0
```

conditional_include_output

`<0 | 1>` Controls whether output pins are included in the `when` condition of hidden power and leakage constructs.
Default: `1` (on)

`0`	Output pins are not included in conditional `when` statements of hidden power and leakage groups.
`1`	Output pins are included in conditional `when` statements of hidden power and leakage groups.

This parameter must be used before the `char_library` command.

Example

```
# no output pins in hidden power & leakage 'when' conditions
set_var conditional_include_output 0
```

conditional_leakage

<0 | 1>	Controls state dependence of leakage. Default: 1	
	0	Disable state-dependent leakage.
	1	Output state-dependent leakage.

This parameter must be used before the char_library command.

conditional_min_period

<0 | 1>	Enables modeling of conditional (state-dependent) min_period. Default: 0	
	0	A single state independent min_period group is written to the output library. If using min_period_when, then a single min_period group is modeled with the when set to the min_period_when.
	1	Write min_period groups to the output library corresponding to all characterized mpw states (see conditional_mpw). If using min_period_when then a min_period group is modeled for each mpw group where the when functionally overlaps with the min_period_when value.

This parameter must be set before the write_library command.

Example

```
# Disable conditional states for leakage
set_var conditional_leakage 0
```

conditional_mpw

`<-1 | 0 | 1>` Controls the output of conditional mpw timing constraints. Default: `-1`

`-1` Follows the `conditional_constraint` setting.

`0` Turns off conditional mpw, regardless of `conditional_constraint`. The mpw is modeled without any state and the MPW low/high is the worst case of all of the MPW values characterized as follows:

```
min_pulse_width_low = max ( all mpwLo )
min_pulse_width_high = max (all mpwHi )
```

`1` Turns on conditional (state-dependent) mpw, regardless of `conditional_constraint`. The mpw is characterized and modeled with state dependence. For a given state and the MPW low/high is the worst case of all of the MPW values characterized for the specific state and is determined as follows:

```
min_pulse_width_low = max ( all mpwLo )
min_pulse_width_high = max (all mpwHi )
```

This parameter must be set to `1` prior to `char_library` to enable the characterization of state-dependent MPW data. It may be reset to `0` prior to `write_library` to disable the modeling of state-dependent MPW.

conditional_rcvr_cap_select_criteria

`<none | first | mid | last>`

Specifies the position in the receiver capacitance table to be used to decide which table to select to write to the output library when there are multiple tables with the same identical "when" condition.
Default: `none`

Sometimes Liberate outputs multiple receiver capacitance tables for the same "when" condition. This usually occurs because the output state is removed from the when resulting in identical states. For example, the states `D&Q` and `D&!Q` are characterized for clock falling hidden power. If the output pin Q is removed from the when, then there is a two capacitance groups with the same when of "D".

Note: When the duplication occurs because conditional_cap_hidden_pin_thresh is exceeded, then duplicate receiver capacitance tables are expected since they correspond to the minimum and maximum input receiver capacitance. The timer expects to see both of these capacitance groups and the parameter should be set to "none".

`none`	No selection is performed. All receiver capacitance tables are written to the output library.
`first`	Selects the table with the largest value in the first index position.
mid	Selects the table with the largest value in the middle of the table. For example, if there are eight points in index_1, compare 4th positional value.
`last`	Selects the table with the largest value in the last index position. For example, if there are eight points in index_1, compare 8th positional value.

Note: This selection is disabled when conditional_cap_hidden_pin_mode is set to 1 and the number of receiver capacitance tables exceeds `conditional_cap_hidden_pin_thresh`. This is because in this case, the timer is expecting to see both the minimum and the maximum capacitance tables.

This parameter must be used after the `char_library` command.

constraint_allow_delay_only_vectors

`<false | true>` Controls the behavior of the tool for finding the constraint vectors and measure constraints for the pins that are logically non-functional.

Default: `false`

`false`	Prevents the tool from finding the constraint vectors and measure constraints for the pins that are logically non-functional.
`true`	Allows the tool to find the constraint vectors and measure constraints for the pins that are logically non-functional.

This parameter must be specified before the `char_library` command.

constraint_async_probe_internal

`<0 | 1>` Specifies whether to probe internal nodes on non-sequential constraints. This parameter only applies to `non_seq_setup` and `non_seq_hold` constraints. (See constraint_probe_internal).
Default: `1` (permits probing internal nodes)

Liberate automatically finds probe nodes for measuring constraints.

`0`	Prevents from choosing an internal node as the probe node for non-sequential constraint acquisition.
`1`	Permits to probe internal nodes.

This parameter must be specified before the `char_library` command.

constraint_bisection_mode

`<0 | 1 | 2 | 3>` Controls the methodology used for bisection searches. Default and recommended: `3`

`0`	Use bisection search.
`1`	Use an earlier implementation of Brent's algorithm. In some corner cases, the results generated using this algorithm can be artificially pessimistic when compared to the ones generated using the pure bisection method. This value should be set only for backward compatibility.
`2`	Use an earlier implementation of Brent's algorithm. This version can unpredictably revert to bisection when it should do interpolation. The result is correct but will not match the correct algorithm precisely and will take more iterations on average. This value should be set only for backward compatibility.
`3`	Use the Brent's algorithm. This algorithm uses interpolation to converge to a solution more rapidly than bisection on average, and it takes no more than twice the iterations of bisection in the worst case.

Note: This parameter controls the primary search sequence. However, due to optimizations done for reducing or increasing the search range where necessary, or for controlling the nominal delay selection, the initial part of the search may vary from the selected algorithm.

In extremely rare cases, setting `1` may result in additional iterations and/or pessimism in the constraint when delay degradation is the success criteria due to numerical noise in the delay measurements. This effect is usually seen if one version of Liberate generates one or two constraint values out of an entire library that are more pessimistic than the `constraint_search_time_abstol` window should allow.

The results for settings `0` and `2` should always be within the `constraint_search_time_abstol` window. The benefit to using `2` is slightly better performance.

To verify a constraint, use `extsim_save_verify=1` to generate a sweep deck, then run in a standalone simulation. Note the nominal delay value and calculate the `delay+degrade` value. Look for the constraint that results in a measured delay that is less than the `delay+degrade` value.

If there are multiple places in the sweep results that cross the `delay+degrade` threshold, than this cell exhibits "multi-bump" or multiple solutions to the `delay+degrade` success criteria. The criteria should be adjusted to result in only one solution for consistency.

This parameter must be used before the `char_library` command.

constraint_check_final_state

<0 | 1 | 2 | 3 | 4> Enforces additional criteria on the constraint measurements. It enables a check on the final state of transitioning nodes in the cell. The final state of internal probe nodes and feedback wires is checked to have fully transitioned. The final voltage on each of the internal probe and feedback wires must agree with their intended final logic level. Checking the final state of these additional nodes provides an assurance that the probe node is stable and will not revert to its original state. The additional node checks normally only affect constraints that use delay degradation as their measurement criteria.
Default and recommended: 4

- 0 — Does not perform extra checks to determine the final state of internal nodes.
- 1 — The circuit simulation continues until each of the internal probe and feedback wires transition to within 5% of their expected final voltage. This setting ensures that these wires reach 95% of their voltage swing.

  Note: This mode is applied only to delay degrade constraints. It does not apply for the nochange constraints with both glitch and delay vectors.
- 2 — The simulation does not stop early. This setting ensures that the probe node transition has fully settled before measuring the degradation by providing extra settling time.

  Note: This mode is applied only to delay degrade constraints. It does not apply for the nochange constraints with both glitch and delay vectors.
- 3 — Same as option 2, but also check for stability of internal feedback wire at the end of the simulation. If any internal feedback node voltage is below constraint_check_final_state_threshold, the iteration is considered a failing bound.
- 4 — Checks the loopback nodes like value 3 does, but excludes the nodes that are known not to transition or will normally end in a floating or conflicting state.

Note: A setting of 2, 3, or 4 may increase a small number of constraint iterations and slightly increase characterization time.

The additional node checks also affect constraints that use glitch-peak criteria where glitches larger than the constraint_check_final_state_threshold are considered as failing measurements.

This parameter must be used before the char_library command.

constraint_check_final_state_threshold

<value>

Sets the final state threshold to be used when constraint_check_final_state is enabled. The threshold value is a ratio between 0 and 1.
Default: 0.9 (use delay_out_fall / rise)

When allowed to default, the threshold uses the delay_out_rise threshold for a rising transition and the delay_out_fall threshold for a falling transition.

This parameter must be used before the char_library command.

constraint_check_rebound

<0 | 1>

Checks for an opposite glitch after constraint probe transition and fails the constraint iteration if detected. The glitch height is the output delay threshold, unless a value is specified with the constraint_check_rebound_threshold parameter.
Default: 0

- 0 — Does not check for probe node reversals that lead to multiple measurements.

1	Reversals on the probe node transition greater than 1pS mark the constraint simulation as failing. If a probe node transition reversal crosses the delay measurement threshold (see delay_out_fall and delay_out_rise) multiple times, then measure the time delta between the first and last threshold crossing. If the delay delta between the two crossings exceeds 1pS, then the bisection bound is treated as a failing bound.

When measuring constraints using delay criteria as the metric, Liberate expects that the probe node transitions from gnd to vdd monotonically. It does not check if the probe node reverses direction. Sometimes the probe node transition reverses course momentarily.

Note: This parameter is not recommended because jitter in the output can exceed the 1pS criteria, leading to unpredictable constraint results. If checks on the output waveform are desired, it is recommended to use constraint_check_final_state and constraint_slew_degrade instead.

This parameter must be used before the char_library command.

constraint_check_rebound_threshold

<value>	Sets the threshold to be used when constraint_check_rebound is enabled. The threshold value can be set to -1, or to a value from 0.0 to 1.0. Default: -1 When constraint_check_rebound is set to 1, setting the constraint_check_rebound_threshold parameter to a threshold value >=0 specifies the fraction of full rail which an output must return to after its initial transition to be considered not to have transitioned. If constraint_check_rebound_threshold is set to -1, the delay threshold is used instead.

This parameter must be used before the char_library command.

constraint_clock_gater

`<0 | 1>` Controls the use of special clock gater circuit constraints.
Default: `1 (enable special checks)`

- `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 Chapter 7, "Performing Characterization using Liberate."

This parameter must be used before the `char_library` command.

constraint_combinational

`<0 | 1 | 2>` Disables combinational measurement of constraints. .
Default: `0`

- `0` Use a bisection search method when characterizing constraints.
- `1` Attempt to use a bisection search, but when that fails, it uses a simple delay-degradation based sweep in an attempt to characterize the constraint.

  Note: This value should be used to characterize a constraint for a combinational cell such as a `nand` gate.
- `2` Enables a path-difference, formula-based constraint estimation if the bisection search fails to characterize the constraint.

Note: Setting this parameter to `1` or `2` can significantly slow down the Liberate run.

This parameter must be used before the `char_library` command.

constraint_combinational_step_limit

<value> — Specify the number of sweep steps.
Default: `1000`

When `constraint_combinational` is enabled (set to 1 or 2), this parameter determines how many sweep steps the constraint characterization can take.

This parameter must be used before the `char_library` command.

constraint_combinational_step_size

<value> — Specifies the desired combinational constraint step size (in MKS units) to be used for linear constraint searches (non-bisection). Also, see constraint_combinational.
Default: `10e-12 seconds`

This parameter must be used before the `char_library` command.

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%)

The set_constraint_criteria command can also be used along with this parameter. If both this parameter and the `set_constraint_criteria` are used, the last one executed sets the value to be used by Liberate.

This parameter must be used before the `char_library` command.

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-to-output-delay or data-to-output-delay and the `constraint_delay_degrade_abstol` is used as the delay degradation criteria.
Default: `2e-12` (2ps)

Note: The minimum supported value is `1e-13` (0.1ps).

Note: For 28nm and below, or for high performance designs the recommended value is `2e-12`. However, for 16nm FinFET and below, the recommended value is `1e-12`.

The set_constraint_criteria command can also be used to set this parameter. If both this parameter and the `set_constraint_criteria` are used, the last one executed sets the value to be used by Liberate.

This parameter must be used before the `char_library` command.

constraint_delay_degrade_abstol_max

<delay> Specifies the maximum 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 minimum of the `constraint_delay_degrade` percentage of the clock-to-output-delay or data-to-output-delay and the `constraint_delay_degrade_abstol_max` is used as the delay degradation criteria. Both `constraint_delay_degrade_abstol` and `constraint_delay_degrade_abstol_max` can be used simultaneously to set both an upper and a lower bound to the delay degradation.
Default: `-1` (not used)

This parameter must be used before the `char_library` command.

Example

```
# use a 10% delay degradation with min=10ps and max=60ps
```

```
set_var constraint_delay_degrade            0.10
set_var constraint_delay_degrade_abstol     10e-12
set_var constraint_delay_degrade_abstol_max 60e-12
```

constraint_delay_degrade_minimize_dtoq

<0 | 1 | 2 | 3>

Sets to use the minimize minimum <setup | recovery> + clk-to-Q delay as the setup and recovery criteria instead of the constraint_delay_degrade criteria. Default: 0 (Do not minimize D to Q delay)

Note: D does not propagate to Q, but minimize D-to-Q is shorthand for minimizing the sum of D->clk and clk->Q.

When this parameter is set to a value other than 0 or 1, the clk to Q delay measurement are simulated with both D and clk switching with the time offset equal to the setup time derived previously.

Important

When using this algorithm, the constraint_delay_degrade_nominal_check parameter should be disabled (=0).
For example:

```
set_var -type {mpw hold}
constraint_delay_degrade_nominal_check 0
```

0 — Use the constraint_delay_degrade criteria when measuring setup and recovery.

1 — The setup measurement threshold for degradation cases is to minimize the D->clk (setup time) + clk->Q delay. Instead of searching for a fixed delay degradation percentage (see constraint_delay_degrade), This method searches for the minimum of D to clk plus clk to Q delays; in other words, minimize the D to Q delay.

2	In addition to the setup measurement method used when set to a 1, the clock to output delay, transition, ECSM, and CCS data is all simulated based on the constrained data switching. After the setup simulations are done, delay simulations are performed and the data is stored in the ldb under rise_constrained_timing and fall_constrained_timing groups.
3	Same as 2, but employs additional calculations to minimize the maximum of setup0 or setup1 (setup to latch a 0 or a 1), plus the maximum for a launch0 or launch1 (output transitioning to a 0 or to a 1).

This parameter must be used before the char_library command.

constraint_delay_degrade_minimize_dtoq_clock_only

<0 | 1>

Controls whether the constraint_delay_degrade_minimize_dtoq algorithm is applied only to clocked constraints (where related_pin is a clock) or to all constraints.
Default: 0 (apply to all constraint arcs).

0	When the constraint_delay_degrade_minimize_dtoq algorithm is enabled, apply the minimize dtoq algorithm to all constraints.
1	Apply the minimize dtoq algorithm only to clocked constraints (the related pin is a clock).

This parameter must be specified before the char_library command.

constraint_delay_degrade_minimize_dtoq_mode

<0 | 1 | 2 | 3 | 4 | 5 | 6 | 7>

Avoids clock to output delay simulations where data switches too early with respect to clock.
Default: 0

0	Use whatever setup time is available.
1	Constraint delay to probe only.
2	Constraint delay for worst setup data only.
3	Constraint delay for proble and worst setup data only.
4	Constraint delay for non-scan setup only.
5	Constraint delay for proble and non-scan setup only.
6	Constraint delay for worst setup and non-scan setup only.
7	Constraint delay for proble, worst setup, and non-scan setup data.

This parameter has an effect only when constraint_delay_degrade_minimize_dtoq is enabled. For example, a DFF (Flip-Flop) has a ck - q arc and assume there are 2 inputs, D and TE that have a setup constraint with respect to CK. When **constraint_delay_degrade_minimize_dtoq** is enabled, the dtoq delay must be evaluated for setup times of both D and TE.

Incase there are four DFF flops (4 bits with D0-D3 and Q0-Q3 and TE). Here the problem is that there are four TE setup times, one for each probe node of Q0-Q3. The worst case TE - CK setup time can come from any probe. In a verbose template for multi-bit cells, it is possible that the setup time only gets measured using one probe node such as Q0. There can be four CK-Qn delays. A problem can occur when the setup time that matches to the specific output pin for each Qn is not measured. A special algorithm is needed to be able to link up specific setup times to specific CK - Qn arcs. For the minimize dtoq algorithm to work correctly, Liberate needs to be certain that the input -CK setup is measured for all four probes or that the worst case probe is used.

This parameter must be specified before the char_library command.

constraint_delay_degrade_minimize_dtoq_tol

<value>

Specifies a relative tolerance (floating point number) to apply to the minimum `dtoq` delay so that setup and recovery times can be reduced, while increasing clock to q delay.
Default: `0` (Do not apply the tolerance)

The total `dtoq` delay is:

```
(1+constraint_delay_degrade_minimize_dtoq_tol) *
min_dtoq.
```

This parameter must be used before the `char_library` command.

constraint_delay_degrade_mode

`<pushout | pullin | pullin_pushout>`

Describes the degradation behavior expected on the probe mode if the metric is delay-degrade.
Default: `pushout`

Note: This parameter can be set on per-arc basis.

`pushout` Expect a normal increase in delay-to-probe as constraint approaches failure.

`pullin` Expect a decrease in delay-to-probe as constraint approaches failure.

`pullin_pushout` Expect a increase or decrease in delay-to-probe, or both, as the constraint approaches failure.

Tip

Alternatively, you set the `-delay_degrade_mode` option with the set_constraint_criteria command.

This parameter must be used before the `char_library` command.

constraint_delay_degrade_nominal_check

<value>

Specifies a positive time offset (in seconds) between the pin and related_pin of a bisection constraint search. This offset (referred to as the *check* setup passing bound) is used for an additional constraint iteration to determine if the probe delay (`Clk->Q` for example) is significantly faster for very large constraint values than the nominal delay found at the constraint bounds. This situation can arise if a floating channel node exists that changes voltage slowly over time and also affects the cell delay. If this condition is detected, the nominal delay is replaced with the faster infinite-setup delay. If the new nominal will result in a solution within the constraint bounds, a warning is issued; otherwise, if not, an error is issued and the search fails.

A drawback to this technique is that it can result in a very pessimistic constraint beyond the one that is required to avoid the meta-stable region and ensure correct cell operation. Additionally, the large offset required to get the fastest delay can result in greatly increased simulation time.

An alternative is to use the delay_constrained_by_setup_recovery parameter to get a delay value consistent with the less pessimistic constraint.

Default and recommended: `0` seconds (disabled)

When this parameter is set to a non-zero value, the following flow is enabled:

1. Determine Dref (the Default probe delay) using the normal constraint bound methodology.
2. Determine Dchk (the check based probe delay) measured by increasing the pin to related_pin offset by the constraint_delay_degrade_nominal_check value.
3. Compare Dref to Dchk using constraint_delay_degrade_nominal_check_abstol and constraint_delay_degrade_nominal_check_reltol criteria. If outside of these tolerances:
   a. If Dchk >= Dref, use Dref as per the normal algorithm (no changes required).
   b. If Dchk < Dref and ((Dchk * (1 + constraint_delay_degrade)) > Dref), then:
      - issue warning message, and
      - set Dref=Dchk and proceed with the bisection search

In other words, if the check based nominal delay is smaller than the reference nominal delay and the degraded check based nominal delay is greater than the reference nominal delay, then take the above action.

c. If Dchk < Dref and ((Dchk * (1 + constraint_delay_degrade)) <= Dref, then:

- issue error message,
- mark the constraint as failed, and
- use mark_failed_data_replacement as the final constraint value.

In other words, if the check based nominal delay is smaller than the reference nominal delay and the degraded check based nominal delay is less than the reference nominal delay, then take the above action.

Note: If backward compatibility parameter constraint_slew_degrade_nominal_check is set to `1`, the `constraint_delay_degrade_nominal_check` parameter will update the delay and slew.

This parameter must be used before the `char_library` command.

constraint_delay_degrade_nominal_check_abstol

<value>	Specifies the absolute tolerance (in seconds) used by constraint_delay_degrade_nominal_check. Default: `0` seconds Recommended: `1e-12` (only if `constraint_delay_degrade_nominal_check` is non-zero)

This parameter must be used before the `char_library` command.

constraint_delay_degrade_nominal_check_reltol

<value>	Specifies the relative tolerance (in `0` to `1` range) used by constraint_delay_degrade_nominal_check. Default and recommended: `0.01`

This parameter must be used before the `char_library` command.

constraint_delay_min_check

<min_time>

Minimum valid reference delay for constraint acquisition.
Default: `-1 seconds` (disables the check)

The total dtoq delay is:

`(1+constraint_delay_degrade_minimize_dtoq_tol) * min_dtoq.`

During constraint acquisition, Liberate finds the nominal `CK -> Q` delay. This reference delay is used as a base for all delay push-out calculations. Due to some circuit design techniques or artifacts of measuring delays at non-threshold voltages (for example, `50%`), the reference delay may be close to `0` or even negative.

Liberate checks the reference delay against the value of `constraint_delay_min_check`, and report an informational message if the reference delay is smaller.

This parameter must be used before the `char_library` command.

constraint_dependent_nominal

`<0 | 1>`

Determines how to compute the reference input-to-probe delay for a dependent constraint.
Default: `0`

`0`	Compute the dependent nominal delay using the margined independent constraint value and maximum dependent constraint value.
`1`	Compute the dependent nominal delay using maximum independent and dependent constraint values. This matches the nominal delay used for the independent constraint

This parameter must be used before the `char_library` command.

constraint_dependent_recrem

`<0 | 1 | 2>` Provides separate control over dependent characterization for recovery and removal timing constraints.
Default: `0` (standard characterization)

For detailed information, see constraint_dependent_setuphold.

`0`	Standard charcaterization of timing constraints.
`1`	Enable recharacterization of recovery timing constraints.
`2`	Enable recharacterization of removal timing constraints.

This parameter must be used before the `char_library` command.

constraint_dependent_setuphold

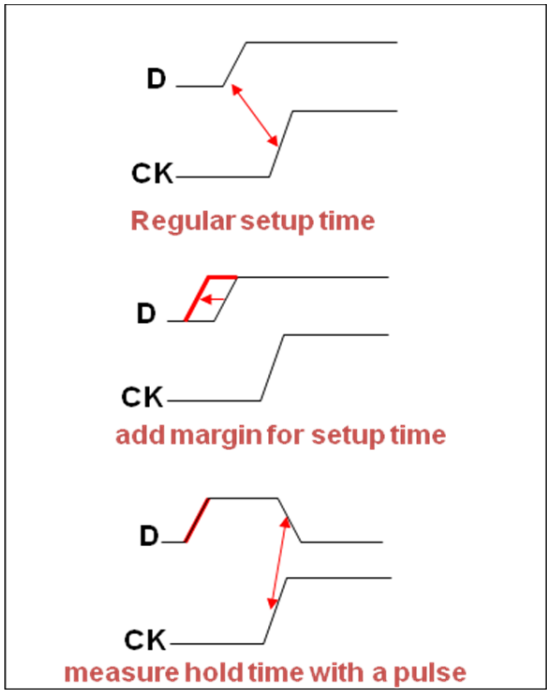
`<0 | 1 | 2>` Enables dependent setup and hold characterization.
Default: `0` (standard characterization)

For detailed information, see Dependent Constraint Characterization.

`0`	Standard standard setup and hold characterization.
`1`	Enable recharacterization of setup.
`2`	Enable recharacterization of hold.

To recharacterize for dependent setup and hold, an ldb must be loaded using `read_ldb` that includes standard setup and hold data.

An example of value `2` (dependent-hold time) is shown in the figure below.



This parameter must be used before the `char_library` command.

constraint_dependent_setuphold_input_threshold

`<value>`

Controls the minimum peak threshold of the input triangular waveform. The acceptable value is a ratio of the supply voltage and is between `0` and `1`.
Default: `-1` (use the value of the control parameters `delay_inp_rise` and `delay_inp_fall` depending on the input transition direction)

When dependent setup/hold is enabled, the data input is a pulse. The input pulse can narrow until it is reduced to a triangular waveform. The peak of this input triangular waveform can drop away from the supply voltage.

This parameter must be set before the char_library command is run.

Example

```
# Set the dependent setuphold input threshold
set_var constraint_dependent_setuphold_input_threshold 0.7
```

constraint_dependent_setuphold_margin

<value> Specifies an absolute margin that is added to the hold (setup) time applied when recharacterizing the setup (hold) for dependent setup (hold) constraints.
Default: 0 seconds

This parameter must be set before the char_library command is run.

constraint_dependent_setuphold_margin_ratio

<ratio> Specifies a margin as the ratio of the `pin_slew` plus the `related_pin_slew`.
Default: `0` (do not apply any input-slew based margin)

This parameter adds a margin as a ratio of the sum of the input slews to the hold (setup) time applied when recharacterizing for dependent setup-hold (see constraint_dependent_setuphold) constraints. The ratio-based margin is determined using the formula specified below:

```
ratio_margin =
     constraint_dependent_setuphold_margin_ratio * ( index_1 +
     index_2 )
```

where:

`index_1` = the input slew of the arc pin

`index_2` = the input slew of the arc related_pin

This parameter must be set before the char_library command is run.

constraint_dependent_setuphold_pessimism

`<0 | 1>` Ensures pessimism over independent setup or hold values during dependent setuphold characterization
Default: `0` (seconds)

`0`	Always use dependent characterized constraint value.
`1`	Report a more pessimistic value between the dependent and independent characterized constraint values

This parameter must be set before the char_library command is run.

Example

```
# Set the output pin name for constraints
set_var constraint_dependent_setuphold_pessimism 1
```

constraint_failed_pin_probe_value

<value> Determines the value of `altos_pin_probe_rise` and `altos_pin_probe_fall` in the LDB.
Default: `0` (in seconds)

This parameter must be set before the char_library command is run.

Example

```
set_var constraint_failed_pin_probe_value 1
```

constraint_failed_related_probe_value

<value> Determines the value of `altos_related_probe_rise` and `altos_related_probe_fall` in the LDB.
Default: `0` (in seconds)

This parameter must be set before the char_library command is run.

Example

```
set_var constraint_failed_related_probe_value 1
```

constraint_failed_value

`1` | *<string>*

Inserts the value of this parameter into the LDB and resulting library for failed constraint data when `mark_failed_data` is not enabled.
Default: `1` (`1e9` in Liberate LDB units of nS)

- `1` — Restores the behavior of 2.5 and prior releases 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.
- *<string>* — 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`.

If the string set by `constraint_failed_value` is like a number (example, `5.5e+09`), then the replacement for the failed constraint value is the result of dividing the number by the time_unit (for example, `1ns`), that is, a scaled number (example, `5.5e+18`) overwrites the failed values in the output library.

This parameter must be set before the `char_library` command.

Examples

Here are some examples for failed constraint replacement.

Example1

```
set_var  constraint_failed_value   "5.5e+09" with a library time unit of 1ns:
fall_constraint (constraint_template_3x3) {
          index_1 ("0.0125, 0.5, 1");
          index_2 ("0.0125, 0.5, 1");
```

```
          values ( \
            "5.5e+18, 5.5e+18, 5.5e+18", \
            "5.5e+18, 5.5e+18, 5.5e+18", \
            "5.5e+18, 5.5e+18, 5.5e+18" \
          );
```

Example2

```
set_var  constraint_failed_value   "inf"
fall_constraint (constraint_template_3x3) {
          index_1 ("0.0125, 0.5, 1");
          index_2 ("0.0125, 0.5, 1");
          values ( \
            "inf, inf, inf", \
            "inf, inf, inf", \
            "inf, inf, inf" \
          );
```

Example3

```
set_var  constraint_failed_value   "NaN"
fall_constraint (constraint_template_3x3) {
          index_1 ("0.0125, 0.5, 1");
          index_2 ("0.0125, 0.5, 1");
          values ( \
            "NaN, NaN, NaN", \
            "NaN, NaN, NaN", \
            "NaN, NaN, NaN" \
          );
```

constraint_glitch_hold

<0 | 1>

Enables glitch height as the failure criteria for characterizing hold time constraints rather than delay degradation.
Default: 0

0	Use delay degradation as the failure criteria. This results in a more pessimistic hold constraint time than probing for a glitch in the output waveform to signify failure.
1	Use glitch height as the failure criteria.

This parameter must be used before the `char_library` command.

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%)

The set_constraint_criteria command can also be used to set this parameter. If both this parameter and the `set_constraint_criteria` are used, the last one executed sets the value to be used by Liberate.

This parameter must be used before the `char_library` command.

constraint_glitch_peak_internal

<value> Specifies the maximum size of logic glitch permitted on an internal node before an arriving signal is deemed to fail a timing constraint (setup, hold, recovery, removal).
Default and recommended: `0.2`

Note: See `constraint_glitch_peak_mode=2` for recommendations for clearing most "Too close to search bounds" errors.

`-1` Use this setting for backward compatibility with the LIBERATE 14.1 ISR3 and prior releases. When set to `-1`, the internal nodes will use the same threshold as constraint_glitch_peak. Therefore, if `constraint_glitch_peak=0.1` (default), then `constraint_glitch_peak_internal` would also use `0.1`.

It is frequently seen that internal nodes have larger glitches than external nodes. Most of these cases are inherent glitches resulting from the momentary contention when a pass-gate is half-open or the circuit is switching between drive paths. In general, these cases represent correct circuit functionality and should be ignored for characterization purposes. The most conservative solution is to slightly increase this parameter to clear these errors. For example, setting `constraint_glitch_peak_internal=0.2` may remove all of the search bound

errors. However, this threshold may vary on a case-by-case basis, and keeping track of all node settings may become tedious.

This parameter must be used before the `char_library` command.

constraint_glitch_peak_max

`<value>` Specifies the maximum threshold for `constraint_glitch_peak_mode`.
Default: `0.5` (Range: 0.0 - 2.0)

If the new threshold from using `constraint_glitch_peak_mode` exceeds the ratio of `constraint_glitch_peak_max` times VDD, then the threshold is limited to the voltage represented by the ratio of VDD specified by `constraint_glitch_peak_max` . This can result in a search bound error.

If there is a rail-to-rail intrinsic glitch (that is, always present) then this can be disabled by setting it to a value greater than 1.0 but less than 2.0.

This parameter must be used before the `char_library` command.

constraint_glitch_peak_mode

`<0 | 1 | 2>` Applies `constraint_glitch_peak` on top of inherent glitch.
Default: `0`
Recommended: `2` (internal node only)

`0` Do not apply `constraint_glitch_peak` on top of inherent glitch.

Note: This value is also used if a constraint arc probes an external pin.

1	Measures the inherent glitch magnitude (noise on the net) and then add that to the constraint_glitch_peak to use as a new threshold. If the new threshold exceeds constraint_glitch_peak_max, the threshold is limited to constraint_glitch_peak_max. This helps prevent warnings about "Too close to search bound" for glitch-based constraint measurements in the Liberate log file. **Note:** This value is also used if a constraint arc probes an internal node.
2	Same as option 1, but only pertains to constraint arcs that probe internal nodes (such as clock gaters). This option is useful when the inherent glitch on internal nodes are larger than the inherent glitch on external pins.

Many cells exhibit inherent glitches immediately upon clock transition. This is a glitch that occurs on a node as a direct result of the clock switching and is not related to any race condition between data and clock. If this inherent glitch occurs at a node that Liberate identifies as the probe node, then the logfile contains the warning message "Too close to search bound". If the constraint measurement criteria is glitch, then it is possible that an inherent glitch occurs on the probe node. Setting constraint_glitch_peak_mode can work around this by accounting for the inherent glitch.

This parameter must be used before the char_library command.

constraint_glitch_peak_report_inherent

<0 | 1>	Enables the reporting (in the log file) of the measured inherent glitch on the probe node. Default: 0 **Note:** The inherent glitch is a glitch that occurs as a result of the related_pin switching and is not caused by race conditions between the pin and the related pin. It is measured only when constraint_glitch_peak_mode is set to 1 or 2.

`0`	Does not report the measured inherent glitch.
`1`	The measured inherent glitch is reported in the log file.

This parameter must be specified before the `char_library` command.

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` parameter.) 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.

This parameter must be used before the `char_library` command.

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 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.

This parameter must be used before the `char_library` command.

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_info_pass_fail

`<0 | 1>` Controls the printing of glitch metric as `"glitch"` or `"pass/fail"`.
Default: `1`

`0`	Print glitch metric as `"glitch"`.
`1`	Print glitch metric as `"pass/fail"`.

This parameter must be specified before the `char_library` command.

constraint_linear_waveform

`<0 | 1>` 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.

This parameter must be used before the `char_library` command.

constraint_margin

`<value>` Specifies to add a margin (in MKS units) to apply when the combinational constraint method is used.
Default: `2e-12` (seconds)

This parameter can be used after the `char_library` command.

constraint_margin_path_delay_backoff

`<value>` Defines a shift from the constraint solution to use in computing path delays for constraint margin calculation. You can specify a positive or negative time in seconds. A negative time can be used to force a failing constraint condition that is needed to get a transition on some internal nodes for hold measurements. A value of `-1` disables this feature.

Use this parameter with the define_arc `-pin_probe -related_probe` command.

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 parameter must be used before the `char_library` command.

constraint_output_load

`<min | max | value | index_?>`

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.
`value`	Specifies an exact load to use.
`index_?`	Specifies the load index value to use from load index on the delay table where `index_0` is the same as `min` and `index_6` is the same as `max`. The load index contains 7 points.

This parameter must be used before the `char_library` command.

Example

Here are some of the examples:

```
# Set the output pin load to 10ff for constraints
set_var constraint_output_load 10e-15

# Set the output pin load to use the fourth load index from the delay table
```

```
set_var constraint_output_load index_3
```

constraint_output_pin

<pin>

Specifies the name of the output pin used for determining timing constraint characterization (setup, hold, recovery, and removal values).
Default: "" (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 parameter defines which of the pins of the sequential cell to monitor.

This parameter may be set to "*". In this case, the constraint searches are repeated on all output pins. The worst-case measurement is reported. Using this functionality can result in approximately a 12% increase in run-time, depending on the number of constraints to be characterized.

Caution

Care should be taken when using constraint_output_pin=* and constraint_info=1 because the output can be misleading. This is because the combination of these parameters reports only the potential probes considered for the constraint. However, it cannot identify the probes that can be rejected as impossible.

This parameter must be used before the char_library command.

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_output_pin_mode

<0 | 1>

Controls the selection method of constraint and MPW probe nodes when using the -io option of the char_library command, if the define_arc command does not specify the desired probe node. In the -io option, it is recommended to specify the desired probe node.
Default: 1

- 0 — Issues an error.
- 1 — Determines the probe node by checking the setting of the constraint_output_pin parameter. If constraint_output_pin is a valid pin or node name, then, that is used. However, if constraint_output_pin is '*', then uses the first output pin in the alphabetical order. Else, issues an error.

This parameter must be specified before the char_library command.

constraint_path_delta_probe_mode

<0 | 1>

Determines the probe threshold when path-delta probe pin is equal to input pin.
Default: 1

- 0 — When path-delta probe pin = input pin, always use a probe threshold of 0.5.
- 1 — When path-delta probe pin = input pin, allow define_arc probe threshold parameters to take effect.

This parameter must be specified before the char_library command.

constraint_probe_internal

`<0 | 1>` Specifies whether to probe internal nodes for constraints.
Default: `1`

The *Inside View* algorithm of Liberate automatically finds probe nodes for measuring constraints. These probe nodes can be primary cell ports or can be internal nodes. This parameter applies to all constraints.

- `0` Prevents Liberate from choosing an internal node as the probe node for constraint acquisition.
- `1` Allows probing internal nodes.

This parameter must be specified before the `char_library` command.

constraint_probe_lower_fall

<value> Probe lower fall threshold.
Default: `0.3`

constraint_probe_lower_rise

<value> Probe lower rise threshold.
Default: `0.3`

constraint_probe_mode

`<0 | 1 | 2 | 3>` Specifies the *Inside View* algorithm for selecting internal probe nodes for constraints. This parameter applies to all constraints.
Default and recommended: `2`

- `0` Selects an output node if possible, otherwise select probes from back-to-back CCCs.
- `1` Selects an output node if possible, otherwise select probes from any internal simulation wire that connects to a transistor gate.

2	Similar to mode 0. However, choose internal node with the preferred metric over an output requiring a different metric.
3	Similar to mode 1. However, choose internal node with the preferred metric over an output requiring a different metric. **Note:** The recommendation is to use setting 2. However, if Liberate fails to find an appropriate probe node with this setting, it results in all constraints of a given type being skipped. In this case, use setting 3.

The patented *Inside View* algorithm of Liberate automatically finds probe nodes for measuring constraints. Output ports (see -output and -bidi options of the define_cell command) nodes are selected first. When no output has an observable change then an internal probe node must be used. Depending on which node is selected to be the probe, Liberate may change the characterization criteria (for example, from delay to glitch peak) if appropriate for the new node. The *Inside View* algorithm has multiple criteria for judging whether or not a node is suitable to be a probe point.

The probe can be set manually using the -probe option of the set_constraint_criteria or the define_arc commands. If a verbose template is available, then manually adding a -probe <node> to the define_arc command for a particular constraint forces Liberate to use that probe node, regardless of the setting of this parameter. The -probe node option of the define_arc command overrides the -probe node option of the set_constraint_criteria command, which in turn overrides the nodes selected by the *Inside View* algorithm.

This parameter must be specified before the char_library command.

Example

```
set_var constraint_probe_mode 2
```

constraint_probe_multiple

`<separate | merge>` Allows for multiple probe nodes in a single constraint search.
Default: `separate`
Recommended: `merge`

Note: For some designs, it may be desirable to test multiple probe nodes in order to find the worst-case constraint. These could be internal nodes or output pins. Combining probe nodes can give a performance increase over separate simulations.

`separate` Uses separate search decks for each probe node.

`merge` Combines all probe nodes in a single search.

Note: In order to enable this feature, the `define_arc` statements need to have multiple probe nodes specified or multiple probes must be specified with `set_constraint_criteria`.

Examples

Example 1: Scan Flip-Flop with two outputs

```
define_arc -type setup -pin D -related_pin CK -probe {Q QN} SDFQNX1
```

Example 2: 4-bit Flip-Flop

```
define_arc -type setup -pin SE -related_pin CK -probe {Q1 Q2 Q3 Q4} MB4X1
```

Example 3: simple Flip-Flop

```
set_constraint_criteria -probe {ILAT Q} -cell DFF
```

This parameter must be used before the `char_library` command.

constraint_probe_upper_fall

`<value>` Probe upper fall threshold.
Default: `0.7`

This parameter must be used before the `char_library` command.

constraint_probe_upper_rise

<value> — Probe upper rise threshold.
Default: `0.7`

This parameter is used to set the voltage thresholds to be used when measuring clock and data delays. For more information, see the `-pin_probe_threshold` and `-related_probe_threshold` options of the `define_arc` command.

This parameter must be used before the `char_library` command.

constraint_search_bound

<min_time> — Specifies the minimum search bound.
Default: `-1`

Use this parameter 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 parameter 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 parameter 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 parameter must be used before the `char_library` command.

constraint_search_bound_bisection_mode

`<0 | 1>` — Determines the definition of a pass/fail for bisection initial bound algorithm.
Default: `1`

The bisection search requires an initial passing search bound and an initial failing search bound.

`0`	The initial pass search bound indicates a transition on the probe node and the initial fail search bound indicates the probe node failed to transition. For example, a flip-flop whose output switches (passing search bound) when data arrives well in front of clock, and fails to switch (failing search bound) when data arrives long after clock.
`1`	The initial pass search bound indicates a transition on the output. The initial fail search bound also indicates a transition on the probe node, but the delay pushout is greater than the `constraint_delay_degrade`. This option is useful when characterizing combinational gates for constraints (see `combinational_constraint`). For example, a `nand` gate is used to gate a clock where the user wants the setup/hold measured between the data and the clock. The output always switches and the delay pushout determines the pass/fail for the bisection search. **Note:** This value is applicable if delay pushout is the criteria. This is because the glitch criteria should already work with the bisection algorithm.

This parameter must be used before the `char_library` command.

constraint_search_bound_estimation_mode

`<0 | 1 | 2 | 3>` Controls the method used to determine the constraint search bound.
Default and recommended: `2`

Note: It is recommended not to change this parameter except for backward-compatibility purpose.

`0` Restores the behavior of the 3.1 and prior releases. The Alspice internal simulator is used to do the search bound estimation. If the internal simulator could not be used, the estimation would not be run.

`1` Restores the behavior of the LIBERATE 12.1 ISR4 and prior releases. This value supports multiple vectors. The Alspice internal simulator is used to do the search bound estimation. If the internal simulator could not be used, the estimation would not be run.

When extsim_exclusive=1 or extsim_model_include is set, a fixed value is used based on constraint_search_bound, if set, or `10ns`.

`2` Uses a method that is consistent for all settings of `extsim_model_include` and `extsim_exclusive`. When this value is set, you can use the desired external simulator for generating SKI estimation.

`3` Disables search bound estimation and uses the value set with the constraint_search_bound parameter.

This parameter must be specified before the `char_library` command.

constraint_search_bound_expand

<integer> — Indicates the number of times the search range should be doubled if the initial bounds fail to bracket a solution. Valid range is from `0` to `9`. Setting this parameter to `0` disables the expansion.
Default: `1`

Note: If the log file contains LIB-52 errors, but the vector is correct, increase the value of this parameter to allow for additional `constraint_search_bound` expansion.

This parameter must be used before the `char_library` command.

constraint_search_bound_probe_mode

`<0 | 1>` — Set this to expand probe selection. Liberate supports any node that is an output of a channel or the input to a gate to be used as a probe node.
Default and recommended: `1`

- `0` — Uses the default available nodes for expanding probe selection.
- `1` — Allows additional nodes such as input nodes connected to pass transistors.

This parameter must be used before the `char_library` command.

constraint_search_iteration_limit

<integer> — Specifies the maximum number of constraint-related bisection search iterations. When this limit is reached, the bisection-based constraint search terminates. The constraint search fails and an error is flagged if the search has not converged within the specified number of iterations. The minimum supported value is 3.
Default: `50`

Note: A value of `25` provides backward compatibility to the Liberate 15.1 ISR3 release.

This parameter must be set prior to the `char_library` command.

constraint_search_mode

`< original | linear_best | linear_test | linear_worst | binary >`

Alters the search sequence to handle multiple constraint solution (double-bump) situations. It enhances bounds estimation to attempt to bracket the metastable region as closely as possible. If the solution is not found within the estimated range, the range is expanded until the solution range is found or an outer limit (controlled by `constraint_search_bound`) is reached. If the solution is in the estimated range, optionally, a coarse linear search for the best or worst solution is performed. If `linear_best` is used, an initial search for a measurement fail boundary is also done to avoid linear search of the hard-fail region. Finally, a binary search is done to achieve the desired precision.

Default: `original`

`original`	Do not change anything and keep the original behavior.
`linear_best`	Perform a linear search of the meta-stable range for the best solution.
`linear_worst`	Perform a linear search of the meta-stable range for the worst solution.
`linear_test`	Similar to `linear_worst`; however, continues the search after the first solution is found to check if multiple solutions exist.
`binary`	Perform a binary search with initial estimation and range expansion as for the linear search options.

constraint_search_time_abstol

<min_time>

Specifies the minimum constraint binary search window. For historical reasons, when constraint_snap_to_bound is set to `1` (default), the `constraint_search_time_abstol` should be set to half of the desired minimum search window (that is, `0.5 * minimum_desired_search_window`).
Default: `2e-12` (seconds)

Decreasing this parameter will increase the run time by increasing the total number of iterations required. This can result in a potentially significant increase in the constraint characterization run time.

This parameter must be used before the `char_library` command.

constraint_search_time_linear_steps

<value>

Specifies the number of linear steps to use in the meta-stable region when constraint_search_mode is set to `linear_best` or `linear_worst`. If the number of steps cause the step-size to be less than `2*constraint_search_time_abstol`, fewer steps are performed with a step-size of `2*constraint_search_time_abstol`.

<value> should be an integer greater than `0`.

Default: `20`

constraint_search_time_linear_threshold

<value>

Provides an estimation of the meta-stable region over which linear search needs to be performed.

<value> is the fraction of the transition at each switching node affecting the constraint. For value x and pins A, B, one side of the meta-stable region is where nodes in response to A reach (1-x) before nodes in response to B reach x. The other side of the meta-stable region is where B nodes reach (1-x) before A nodes reach x. Larger values result in a narrower linear search range. An aggressive value will not cause a constraint failure because the search will expand as needed to find a solution; however, it will reduce the range over which a linear search will be considered, increasing the possibility that the solutions exist outside that range.

<value> should be a floating point value between `0` and `0.5`.

Default: `0.2`

constraint_slew_degrade

<value>	Includes slew degradation percentage along with delay degradation as a criteria for determining timing constraints. Default: `-1`(only use delay degradation)

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 a value (`0.0 to 1.0`) that represents the percentage of slew degradation and is measured using the `measure_ slew_*` parameters.

The set_constraint_criteria command can also be used to set this parameter. If both this parameter and the `set_constraint_criteria` are used, the last one executed sets the value to be used by Liberate.

This parameter must be used before the `char_library` command.

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
```

constraint_snap_to_bound

`< 0 | 1 >` Determines how Liberate selects the bisection constraint value.
Default: `1`

- `0` Liberate uses a proprietary algorithm to determine the next bisection search time. An offset time is chosen between the last passing and failing iterations for a passing bound. If the resulting search window is less than the constraint_search_time_abstol, the bisection search reports the offset time as the constraint value.

  The bisection run time is reduced by one iteration. This final offset time is not simulated.
- `1` Liberate bisection-based constraint characterization will snap to the last simulated passing search bound while characterizing constraints. This setting provides greater confidence in the reported constraint and can increase run time. By reporting the last passing bound, it is reassuring that the reported constraint value was actually simulated.

This parameter must be used before the `char_library` command.

constraint_sweep_pulse_detection_mode

`< 0 | 1 >` Enables a sweep based algorithm instead of the bisection algorithm. The sweep algorithm can significantly increase the characterization runtime.
Default and recommended: `0`

- `0` Use the bisection algorithm to characterize constraints.

`1`	Assumes the constraints exhibit a bump or pulse profile of the delay degradation at the output instead of a single valid constraint value. When set to `1` the constraint is acquired using a sweep algorithm. You can provide the expected minimum width of the passing pulse (bump) using the parameter constraint_sweep_pulse_width_max. The minimum step (accuracy resolution) is controlled by the parameter constraint_combinational_step_size.

Some cell constraints exhibit what is commonly referred to as a double bump. This occurs when a constraint has `2` legal constraint values. For example, in a DFF (D-Flip-Flop), as the data sweeps setup time toward the clock, the failure criteria is exceeded and a setup time is found. However, if the data continues forward, the circuit starts to pass the criteria and eventually again fails the criteria.

This double bump is sometimes related to re-convergence or coupling issues in the circuit design. The default constraint characterization algorithm in Liberate is based on a bisection search. If the constraint exhibits a double bump (aka a constraint pulse), then the bisection search can report either of the setup times.

This parameter must be specified before the `char_library` command.

constraint_sweep_pulse_width_max

<value>	Specifies the maximum width of a constraint pulse (double bump) in seconds. Liberate uses a step size `2.5` times smaller than this value to ensure hitting the bump. The constraint_sweep_pulse_detection_mode parameter must be set to enable the sweep pulse algorithm.

This parameter must be specified before the `char_library` command.

Example

```
set_var constraint_sweep_pulse_width_max 20e-12  # (20pS)
```

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 parameter specifies a duration in seconds to add to the transient simulation end time (see `.tran` in a SPICE format simulation deck). Liberate checks if a transition on the constraint probe occurs within `0.2*constraint_tran_end_extend` 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 parameter. 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 parameter must be specified before the `char_library` command.

constraint_tran_end_extend_retry

<value> — When set to greater than `0`, provides an alternate value for `constraint_tran_end_extend` when Liberate detects a condition where the simulation time may be too short to properly detect a glitch.

When the value is specified as `0`, this feature is disabled and no retry is done.
Default: `0.001`seconds

constraint_tran_end_mode

`< 0 | 1 | 2 >` Enables a methodology for setting transient simulation end time in constraint bisection searches when using various simulators. Default and recommended: `1` (Optimized per table entry)

Note: SPICE simulators have different capabilities and limitations.

`0`	This value is used for backward compatibility only. You can use mode 2 for SPECTRE, SKI, and some proprietary simulators; otherwise use mode 1.
`1`	Uses an optimized `tran_tend` (transient simulation end time) for each entry of each iteration except for some proprietary simulators, which uses mode 2.
`1`	Uses the maximum `tran_tend` for each entry in the constraint data table for all simulations. Use this value only when needed and usually only for testing purposes. This setting may increase runtime.

This parameter must be used before the `char_library` command.

constraint_user_defined_probe_mode

`< 0 | 1 >` Enables to use only user-defined probes for setup/hold, recovery/removal.
Default: `1`

`0`	Behavior of release 12.1.1 and earlier where Liberate treats the user-defined probes as a suggestion.
`1`	Liberate uses only user-defined probes for setup, hold, recovery and removal (this is already the behavior for MPW). User-defined probes are specified with the `-probe` option of either `define_arc` or `set_constraint_criteria`.

This parameter must be specified before the `char_library` command.

constraint_vector_equivalence_mode

`< 0 | 1 >`

Enables an algorithm that determines which vectors need to be simulated for constraints and which can be skipped as equivalent to the simulated vectors.
Default: `0`
Recommended: `1`

`0` Uses base algorithm for reducing vectors based on matching all switching modes

`1` Enables a vector equivalence check that helps reduce the number of vectors simulated when not using mega-mode.

In the case of complex multi-register cells, there can be many delay paths active besides the ones relevant to the particular measurement. For instance, when measuring a constraint at one element of a latch bank, the behavior of the other latches is usually not significant. Previously, Liberate required that the switching activity of all nodes be the same before two vectors could be considered equivalent. With this algorithm, nodes that do not contribute to the measured delay paths are not considered in the vector comparison. This can result in a substantial reduction in the number of simulations required.

This feature is not required when using mega-mode (`mega_enable=2` or the `-type` option of `define_cell` is `mega`) since that mode performs the same function more efficiently. Therefore, `constraint_vector_equivalence_mode` is disabled.

This parameter must be used before the `char_library` command.

constraint_vector_mode

`< 0 | 1 | 2 | 3 | 4 >`

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 re-characterization 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.
3	Enables simplified prioritization of probe or vector selection for constraints. This selection method considers the following aspects in the sequence given below: **1.** The `-probe` option of the define_arc command **2.** The `-metric` option of the `define_arc` command **3.** The default preferred metric for the constraint type **4.** The probe ordering as controlled by constraint_probe_mode

This selection method gives a more predictable behavior for complex cell setup/hold, and more flexibility in honoring the user-defined recovery/removal and non-sequential arcs.

Important

Prior to Liberate 16.1 ISR4, in some situations, setting `constraint_vector_mode` to `3` could give incorrect results when used with `constraint_probe_mode` set to `0` or `1`. Therefore, setting `constraint_probe_mode` to `2` (default) or `3` was required. From Liberate 16.1 ISR4 onwards, the problem has been resolved to allow `constraint_vector_mode=3` to work correctly with all settings of `constraint_probe_mode`, including `0` and `1`.

4 — Considers all vectors at the selected probe that can be measured with degrade or glitch metric.

Note: If a metric is specified using the define_arc or set_constraint_criteria command, then only the vectors that can use the specified metric are considered.

This parameter must be specified before the `char_library` command.

constraint_vector_mode_compare

`< -1 | 0 | 1 | 2 | 3 | 4 >`

Enables debugging messages against the specified setting of constraint_vector_mode.
Default: `-1` (Disabled)

Note: The `constraint_vector_mode_compare` parameter supports the same `0` to `4` settings as the constraint_vector_mode parameter.

Set `constraint_vector_mode_compare` to a value greater than or equal to `0` and ensure that this value is different from the value set using `constraint_vector_mode`. This enables a comparison of the vector and probe selection results that are generated considering the `constraint_vector_mode` and `constraint_vector_mode_compare` settings, and then reports the differences if any are found. There is some preprocessing cost to doing this. Therefore, the default is to disable the comparison.

This parameter must be set before the `char_library` command is run.

Examples

Example 1

```
# Characterize with constraint_vector_mode=3
# Compare vectors against constraint_vector_mode=2
set_var constraint_vector_mode 3
set_var constraint_vector_mode_compare 2
```

Example 2

```
# Characterize with constraint_vector_mode=2
# Compare vectors against constraint_vector_mode=3
set_var constraint_vector_mode 2
set_var constraint_vector_mode_compare 3
```

constraint_width_degrade

<value>

Specifies the percentage of degradation in the width of a pulse when calculating setup/hold time at the output or internal node of the pulse generator in pulse latch cells. This parameter can be used with the `-metric width` option of `define_arc` command.
Default: `0.1` (10%)

constraint_width_degrade_abstol

<value>

Specifies the minimum pulse width degradation value permitted (in seconds) at the output or internal node of the pulse generator in pulse latch cells when measuring timing constraints (setup, hold, recovery, removal) with `define_arc -metric width` and `constraint_width_degrade`. The maximum of the `constraint_width_degrade` percentage of the nominal pulse width and the `constraint_width_degrade_abstol` is used as the width degradation criteria.
Default: `5e-12` (5ps)

constraint_width_degrade_abstol_max

<value>

Specifies the maximum width degradation value (in seconds) permitted at the output or internal node of the pulse generator in pulse latch cells when measuring timing constraints (setup, hold, recovery, removal) with `define_arc -metric width` and `constraint_width_degrade`. The minimum of the `constraint_width_degrade` percentage of the nominal pulse width and the `constraint_width_degrade_abstol_max` is used as the width degradation criteria. The `constraint_width_degrade_abstol` and `constraint_width_degrade_abstol_max` parameters can be used simultaneously to set an upper and a lower bound to the delay degradation.
Default: `-1` (not used)

constraint_worst_vector_abstol

<min_time> Enables this parameter to control minimum constraint binning. Liberate performs simulation-based vector binning when an arc contains multiple vectors. The worst vectors are selected based on results falling within `constraint_worst_vector_abstol` of the worst vector. It is recommended to set this parameter to the same value as `constraint_search_time_abstol`.
Default: `0` (do not apply an absolute tolerance)

This parameter must be used before the char_library command is run.

cpu_load_threshold

<value> Specifies the threshold of average CPU load. If the load exceeds the specified value, the client will be shut down. You can specify a value between `0` to `1`. Default: `0.9`

cpu_memory_min

<double> Specifies memory, in GB, that should be free at a given point of time. The job being processed at the client is stopped if the free memory goes below the specified value. Default: `1`

cpu_memory_rel

<double> Specifies the percentage memory that should be free at a given point of time. The job being processed at the client is stopped if the free memory goes below the specified value. Default: `1`

datasheet_truthtable_in_pin_limit

<value> Sets the maximum number of inout (bidi) pins allowed in a cell to write truth table for the cell in the datasheet. This means that if this parameter is set to 8 and a cell has more than 8 input or inout pins, truth table will not be written for this cell in datasheet.
Default: `8`

The minimum value for this parameter is `0`.

The maximum value is the largest supported integer. However, setting a large value will impact the run time and might make the datasheet generation process run for an indefinite time.

The specified value should depend on the number of input/bidi pins in the circuit. It is not realistic to set this parameter to a value any larger than the total number if `in_pins` (that is, input + bidi pins).

This parameter must be used before the write_datasheet command is run.

Example

```
read_library my.lib
set_var datasheet_truthtable_in_pin_limit 20
write_datasheet -format html -dir tt_html  tt
```

def_arc_drive_side_bidi

`< 0 | 1 >`

Enables to drive bidi pins that are not the pin or `related_pin` of an arc.
Default: `1`

`0`	Does not to drive side pins that are bidirectional.
`1`	Drives any bidirectional pin that is not the pin or the `related_pin` of an arc.

This parameter must be specified before the char_library command is run.

def_arc_msg_level

`< 0 | 1 >` Instructs Liberate to report `define_arc` commands that are not characterized.
Default: `1`

- `0` Reports an error when no valid vectors for the arc are found.
- `1` Reports a warning for every `define_arc` command that is *not* characterized.

Note: The `write_template` command is enhanced to add *"set_var def_arc_msg_level 0".* This change ensures that all new template files created from `write_template` follows the behavior prior to release 3.2.

This parameter must be specified before the `char_library` command.

def_arc_vector_consistency_check

`< 0 | 1 >` Switches to specify the "when" state must agree with the pre-switching state of the `-vector`.
Default: `1`

- `0` Sets the behavior of release 3.1p4 and earlier.
- `1` The "when" state must agree with the pre-switching state of the `-vector`.

This parameter must be specified before the `char_library` command.

default_power_avg_mode

`< 0 | 1 >` Enables a weighted average of all `internal_power` states under a pin.
Default: `1` (weighted average)

- `0` Uses simple average.
- `1` Uses weighted average. When weighted average is used, the `default_power` is computed as though `conditional_expression` was set to separate.

When **conditional_expression** is set to `merge`, the number of `internal_power` groups under a pin can be reduced due to merging of states. When this happens and an average default power group was requested, an average of the groups under the pin is used. The default `internal_power` is computed as a simple average or a weighted average of all of the available states.

This parameter must be used before the `write_library` command.

default_rcvr_cap_groupwise

`< 0 | 1 >` Sets the default method of generating receiver capacitance tables.
Default: `0`

- `0` Generates the default `ccs_receiver_capacitance` group by selecting each `ccs_receiver_capacitance` piece independently based on the heuristics from all matching timing groups.
- `1` Selects the advanced receiver capacitance tables from the same timing group.

This parameter must be set before the `write_library` command.

define_arc_ignore_mode

`< 0 | 1 | 2 >`

Ignores all arcs that satisfy the `when` condition for the `-ignore` option of the `define_arc` command and when `define_arc_ignore_mode` is set to `1`. This parameter allows similar fine-grained control for the `define_arc` command. Default: `1`

- `0` — Ignores all arcs for the pin and `related_pin` if the `-ignore` option of the `define_arc` command is used, All other `define_arc` options are also ignored.
- `1` — Considers `when` condition in the `define_arc` command in addition to `0`.
- `2` — Allows to ignore arcs of type power for the specified pin or `related_pin`.

The *Inside View* algorithm uses the information given in the `define_cell` command as a starting point for determining a cell's functionality. The default behavior of Liberate is to find all vectors possible from this general description.

In cases where the design intent is different from the circuit behavior, there are additional controls to restrict the vectors that *Inside View* algorithm generates. The `define_cell` command accepts a `-when` option which, when used with `"set_var simultaneous_switch_from_cell_when 1"` restricts the full vector set to only vectors that satisfy the `when` condition.

Example

```
define_arc \
-ignore \
-type power \
-related_pin CDN \
-pin Q \
DFF
define_arc_ignore_mode
```

This parameter must be used before the `char_library` command.

define_arc_preserve_when_string

`< 0 | 1 | 2>` Enables to preserve a `when` string in its original form and output to the library.
Default: `0` (off)

Normally, Liberate attempts to simplify a `when` condition according to Boolean logic. For example, in the `define_arc -when "A&B | A&!B"` command, Liberate could simplify the `when` condition to `-when "A"`.

- `0` — Allows Liberate to simplify the `when` string.

  If conditional_expression is set to `separate`, the `when` string is split, but `define_arc_preserve_when_string=0` does not simplify the Boolean logic.
- `1` — Preserve the `when` string in its original form and output to library.
- `2` — Improves over the results that are generated if `define_arc_preserve_when_string=0` where the `when` string will be split if `'conditional_expression=separate'`, but it would not simplify the Boolean logic.

define_cell_missing_template_action

`< error | warning >` Specifies how Liberate should handle a missing template in the define_cell command.
Default: `warning`

- `warning` — A warning will be issued and Liberate will continue to characterize the cell. Power is characterized even if the power template is missing and is modeled with template name `"missing_powet_lut..."` that will not compile.
- `error` — An error will be reported if the `define_cell` command encounters a missing template or mismatched template type. The cells with a bad template will be skipped.

This parameter must be used before the first define_template command.

define_duplicate_cap_mode

`< 0 | 1 >` Specifies the aliasing of pins for the determination of pin capacitance.
Default: `0` (off)

- `0` — No pin aliasing for pin cap characterizations; each pin cap will call for a separate simulation to characterize its value.
- `1` — Use aliasing to determine pin capacitance; a given pin will be characterized, and its value is applied to all the other pins specified with the `define_duplicate_pins` command.

This works together with the define_duplicate_pins command to specify which pin is used to characterized the pin capacitance, and then applied to the duplicate (aliased) pins.

This parameter must be used before the `char_library` command.

define_input_waveform_check_action

`<0 | 1>` Specifies the action to be taken when user-defined waveforms (see define_input_waveform) are specified and one or more input slews are missing in a waveform. User-defined waveforms are added to the template file when the original library has Normalized Driver Waveforms (NDW) and the write_template `-driver_waveforms` option is used.
Default: `0`

- `0` — Reports a warning and creates a piece-wise linear (PWL) waveform by interpolation of an existing waveform.
- `1` — Creates a linear ramp waveform instead of an interpolated PWL waveform and marks the cell as failed.

This parameter must be used before the `char_library` command.

delay_constrained_by_setup_recovery

`< 0 | 1 >` Controls how timing constraints are characterized.
Default and recommended: `0`

- `0` Characterizes the clock to output timing using all slews and loads with infinite setup.
- `1` Characterizes the timing constraints with the normal delay degradation method, but clock to output delays are measured with the data pin also switching – at a time offset determined by the measured setup/recovery time.

  Also, the `constraint_delay_degrade_minimize_dtoq` parameter is forced to 0.

This parameter must be used before the `char_library` command.

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_min_max_mode

`< 0 | 1 >` Controls whether multiple vectors should be characterized for a single delay arc.
Default: `0`

- `0` Disables the feature.
- `1` Characterizes each delay twice, one by selecting the min vector and the next by selecting the max vector. Both these values are then available for default group selection enabled through the following setting:

```
set_default_group -criteria {delay min_max}
```

This parameter must be used before the `char_library` command.

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 parameters set the input and output rising and falling transition crossing-points for measuring delays. It is possible to modify the delay measurement thresholds after characterization. To modify these thresholds, use the `read_ldb` command, then modify the required delay-measurement thresholds (see delay_inp_fall, delay_inp_rise, delay_out_fall, and delay_out_rise) and write the new library.

Note: If the library is characterized with a predriver cell, there may be a small accuracy impact when changing the measurement thresholds without re-characterizing. Only combinational rise/fall arc-related delays can be modified in this way. Timing constraints cannot be modified without rerunning the characterization

These parameter must be used before the `char_library` command.

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
```

disable_method

<0 | 1 | 2>

Controls how the timing arcs for tri-state devices are calculated when the device is transitioning to the `OFF` state (`three_state_disable`). See the figure below.
Default: `0`

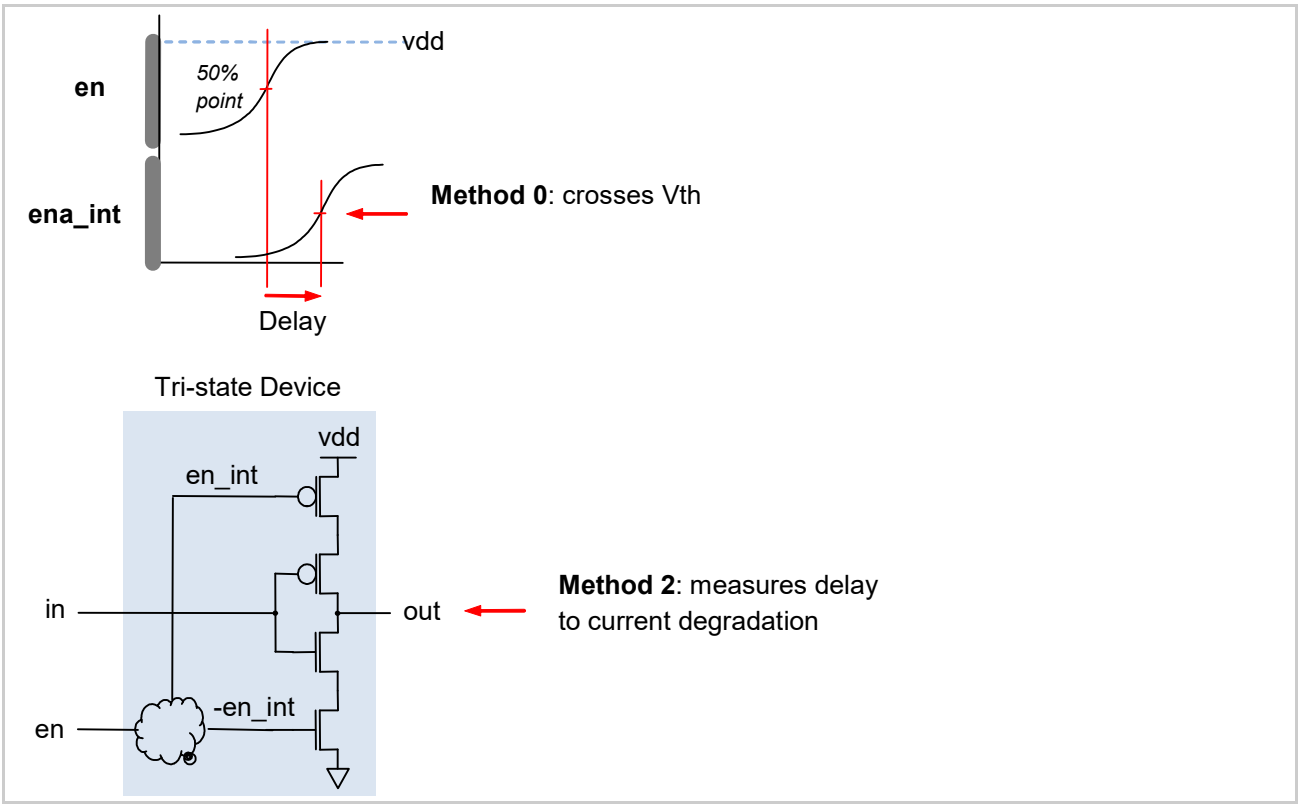
`0` Calculates the disable timing arcs (`three_state_disable`) by measuring the delay time from the input disable pin to the time the input signal to the last transistor stage (channel connected block) crosses the lowest threshold voltage (V_{th}) of the impacted transistors.

`1` Measures the delay from the enable pin (en) crossing the 50% threshold, to the input pin of the channel-connected region (`en_int` or `-en_int`, whichever is worse) reaching half the supply voltage (Vdd/2)

`2` Measures the delay from the enable pin (en) crossing the 50% threshold, to the time when the output drive current degrades to 10% of the short-circuit current.

Note: This method is automatically enabled if `disable_method` is set to something other than `2` but Liberate fails to find any last stage CCR input to probe. However, if the user specifies an internal probe, Liberate respects that regardless of the `disable_method`.

Figure 5-1



Note: The `timing_sense` for the `three_state_enable` or `three_state_disable` arc may have a different meaning from the default one. According to the Synopsys Liberty specification, if a value of `1` on the control pin of a three-state cell causes a `Z` value on the output pin, the `timing_sense` is `positive_unate` for the `three_state_disable` timing arc and `negative_unate` for the `three_state_enable` timing arc. If a value of `0` on the control pin of a three-state cell causes a `Z` value on the output pin, the `timing_sense` is `negative_unate` for the `three_state_disable` timing arc and `positive_unate` for the `three_state_enable` timing arc.

This parameter must be used before the `char_library` command.

Example

```
# Measure disable delay to Vdd/2
set_var disable_method 1
```

discard_timing_sense_after_merge

`< 0 | 1 >` Enables removal of `timing_sense` after the merging of two timing groups.
Default: `0` (Do not remove `timing_sense`.)

- `0` Does not remove `timing_sense`.
- `1` Removes the `timing_sense` after the merging of two timing groups.

See timing_group_unateness.

This parameter must be set before the `char_library` command.

disk_wait_time

<value> Specifies the number of seconds you want Liberate to wait before it attempts to read the external simulator output file (`sim.lis`). When Liberate is using an external simulator (see `char_library -extsim`), Liberate attempts to read the simulator output file as soon as the simulator terminates. Due to network latencies, this file may not yet be ready to read.
Default: `0` (seconds)

This parameter must be used before the `char_library` command.

driver_cell_acc_mode

`< 0 | 1 >` Generates a more accurate driver waveform for minimum slews.
Default: `1` (on)

- `0` Restores the behavior of the 3.1p2 and earlier releases in sampling active driver waveforms.
- `1` Generates a more accurate driver waveform for the minimum slews.

This parameter must be used before the **char_library** command.

driver_cell_all_inputs

`< 0 | 1 >` Holds side pin constant with driver cell.
Default: `0`

- `0` When a driver cell is specified for the cell/pin, and the pin is a constant side input connected to a transistor channel (source/drain), then the side pin is held to a constant value by the specified driver cell.
- `1` Same as setting `0` except that all constant side pins will be driven by the specified driver cell.

This parameter must be used before the `char_library` command.

driver_cell_info

`< 0 | 1 >` Enables reporting for each cell input. The report includes the driver cell that is the characterized driver for the input. This report is useful to determine the driver cell assigned to each input when many `set_driver_cell` commands are used.
Default: `1`

- `0` Does not report for each cell input.
- `1` Requests a report for each cell input.

This parameter must be used before the `char_library` command.

driver_cell_load_all_outputs

`< 0 | 1 >` Enables reporting if the active driver load is applied to secondary (side) pins while adjusting loads to match slews on primary pins. For this functionality to work, the set_driver_cell `char_pin` and `pin_map` options must be used to map active driver pins to cell pins.
Default: `1` (apply the load to the secondary side pins)

- `1` Applies the load to the secondary side pins.

This parameter must be used before the `char_library` command.

driver_cell_load_ldb_cmd

`< 0 | 1 >` Instructs Liberate to honor all `set_driver_cell` commands stored in an ldb that is loaded using the `read_ldb` command. The presence of the `set_driver_cell` commands is required to enable the output of Normalized Driver Waveforms (NDW) into the library when starting from an ldb.
Default: `0`

This parameter must be used before the `read_ldb` command.

driver_cell_trim_miller

`<0 | 1 | 2>` Enables filtering of non-monotonic behavior from the waveform.
Default: `0`

When an active driver is used (see set_driver_cell) to create an input waveform, the waveform may be non-monotonic due to effects such as the miller capacitance. In this scenario, use the `driver_cell_trim_miller` parameter to filter the non-monotonic behavior.

- `0` Do not apply any filtering. Use the waveform produced by the active driver.
- `1` Apply filtering to remove non-monotonic behavior from the input waveform. This setting should only be used for backward compatibility to the LIBERATE 16.1 ISR4 and prior releases to match existing libraries.
- `2` At times, the active driver cell is not able to produce a waveform at the fast slews. In these cases, Liberate adjusts the fastest slew that the driver can create to meet the required fast slews in the slew index. This setting ensures creation of a proper waveform in the rare cases where the active driver is not fast enough. If filtering of non-monotonic behavior is desired, the setting of `2` is recommended.

This parameter must be set before the `char_library` command is run.

driver_type_model_pad_check

`< 0 | 1 >`

Adds a check for pad pins when Liberate outputs the `driver_type` attribute for cell output pins. This check is useful to prevent the `driver_type` attribute from being modeled for tie-high and tie-low cell outputs.
Default: `0`

- `0` — Do not check for pad pins such as `is_pad` when deciding to output the `driver_type` attribute.
- `1` — Models the tie-high and tie-low cells with the `driver_type` attribute for only pad pins.

This parameter must be specified before the `write_library` command.

driver_waveform_arcs_only

`< 0 | 1 >`

Enables to select the method to be uses to filter the input driver waveforms that are stored and available for output into the `.lib` file.
Default: `0`

- `0` — Writes out all of the waveforms that are used during characterization into the output library without any regard to the type of data the waveform was used to characterize.
- `1` — Enables to store only those input waveforms used to characterize arcs for delay and power. If there are no delay or power arcs, then no input driver waveform data is stored. When the library (or pin) is characterized only for input capacitance (see char_library `-skip`), the library database does not have any characterization data from delay or power tables to save.

  Note: Use this setting to restore the behavior of the 2.4p2 and prior releases.

This parameter must be specified before the `write_library` command.

driver_waveform_output_precision

`<value>` — Specifies the number of digits of precision to maintain when generating driver waveform data.
Default: `0`

The default setting of zero allows Liberate to select the number of digits to be used.

This parameter must be specified before the `read_library`, `read_ldb`, or any other command that generates driver waveform data. This includes `write_template` and `write_library`.

Example

```
set_var driver_waveform_output_precision 10
read_library my.lib
write_template template.tcl
```

driver_waveform_pulse_mode

`<0 | 1>` — Enables to choose the pulse algorithm. When combining leading and trailing driver waveforms to form a pulse, sometimes a sharp spike could occur depending on the time delta between the last point chosen on the leading waveform and the first point chosen on the trailing waveform.
Default: `1`

- `0` — Restores the behavior of the 2.4 and prior releases.
- `1` — Creates a new point to approximate where the two waveforms cross.

This parameter must be used before the `char_library` command.

driver_waveform_slew_index_tolerance

<value>

Controls filtering of the driver waveform indexes. If it is set as `1e-12(1ps)`, any driver waveform index that is less than `1ps` to the last index would be ignored.
Default: `1e-12` or `1e-15` (`1e-15` is for `7nm` or smaller characterization)

The value is specified in seconds. Any positive real value is acceptable, but a reasonable value is `1e-12` (`1ps`) or smaller `1e-15` (`1fs`).

For best performance, this parameter is set to `1ps` (default). However, when it is related to a `7nm` or smaller library, ensure that this parameter is set to `1fs` to get correct results.

This parameter must be used before the write_library or append_library command is run.

Example

```
set_var driver_waveform_slew_index_tolerance 1e-15
```

driver_waveform_wildcard_mode

`<0 | 1>`

Enables to control formatting of the Normalized Driver Waveform (NDW) data in the output library when the define_input_waveform command is used to specify the input waveforms and the pin names specified with it contain wildcards,
Default and recommended: `0`

`0`

Wildcards in the `define_input_waveform` command are not supported when writing `NDW` data in the output library. If wildcards exist in a `define_input_waveform` command, `NDW` is not written out. In addition, it is recommended that wildcards are not used in the `define_input_waveform` command.

1	Wildcards in the define_input_waveform command are expanded when writing NDW data into the output library. This can lead to many large NDW tables in the output library.

This parameter must be used before the char_library command.

duplicate_pin_attr_mode

<use_master | augment_master>

Controls how master pin attributes are populated to duplicated pins.
Default: use_master

use_master	All attributes from the master pin are copied to the duplicate pin except those generated from characterization, that is, pin capacitance.
augment_master	Copy all the attributes from the master pin to duplicated pin, but do not overwrite attributes that have already been set for that pin.

duplicate_risefall_power

< 0 | 1 >

Enables power duplication.
Default: 0

Note: In the Liberate MX and AMS flows, duplicate_risefall_power is set to 1 by default for backward compatibility in the regression.

0	The missing power direction is represented by a scalar group with value of 0.

`1`	Duplicates the power values when only rise or fall power (but not both) exist, For example, if only *rise_power* exists but not *fall_power*, then copy the *rise_power* values into *fall_power*. This duplication only applies to *internal_power* groups where the output pin is switching. It does not apply to hidden power.

This parameter can be used after the `char_library` command.

Example

```
# Duplicate existing power to missing power
set_var duplicate_risefall_power 1
```

duplicate_risefall_power_ccsp

`<0 | 1 | 2>` Controls copying rise and fall CCSP dynamic currents into a missing rise and fall group.
Default: `0`

`0`	Does not copy the rise and fall CCSP dynamic currents.
`1`	Copies rise and fall CCSP dynamic currents into the missing rise and fall group (similar to what the `duplicate_risefall_power` parameter does).
`2`	Same as setting it to 1, except Liberate includes the `input_switching_condition` in the copied group.

This parameter must be used before the `char_library` command**.**

ecsm_arctype_enable

`<0 | 1>` Enable modeling of the `ecsm_arctype` attribute in ECSM format libraries.Default: `0`

Note: In the Liberate MX and AMS flows, `duplicate_risefall_power` is set to `1` by default for backward compatibility in the regression.

- `0` Does not output the `ecsm_arctype` attribute in ECSM format libraries.
- `1` Enable modeling of the attribute `ecsm_arctype` for `ecsm_vivo_current_waveform` groups in ECSM noise model. The optional `ecsm_arctype` attribute is a string with four valid values: `max_rise`, `max_fall`, `min_rise`, and `min_fall`. The `max_rise` and `max_fall` values represent the ViVo arcs corresponding to the slow transition of the signal on the output. While `min_rise` and `min_fall` are for the fast transition. Including this attribute may help to clear up messages when using the `ecsm_checker` utility.

This parameter must be set before the `char_library` command.

ecsm_cap_hidden_pin

`<0 | 1 | 2>` Specifies how ECSM capacitance is written for input pins. Default: `2`

Prior to version 3.1, the `ccs_cap_hidden_pin` parameter (default = 1) controlled how the ECSM capacitance was written. The `ecsm_cap_hidden_pin` parameter provides a separate control for this.

- `0` Hidden pins only.
- `1` Hidden and half-hidden pins. *(Set this for pre-3.1 behavior.)*
- `2` All input pins.

This parameter must be used before the **write_library** command.

ecsm_cap_input_slew_mode

`<0 | 1>` Enables ECSM capacitance input slew adjustment mode. Default: `0`

- `0` Writes the original `index_1` for `ecsm_capacitance`. In this mode, the `index_1` slew values in ECSM data tables are not adjusted.
- `1` Use the adjusted `index_1` slew values from release 3.1p2 and earlier.

  Note: This adjustment is no longer recommended.

This parameter must be used before the **write_library** command.

ecsm_cap_load_effect_tol

`<value>` Specifies the `ecsm_capacitance_set` effective threshold. Default: `0.001 (0.1%)`

If an arc, for example, `Clk -> Q`, has load dependent ecsm capacitance values that are within the tolerance limits, it means that the pin is shielded from the output load. The verification is done for each arc. When the load effect is less than this tolerance, then output the average cap over all the loads.

This parameter specifies the tolerance used when the ecsm_cap_style parameter is set to `capacitance_set`.

This parameter must be set before the `write_library` command.

ecsm_cap_mode

`<0 | 1 | 2>` Sets the ECSM capacitance modeling mode.
Default: `2` (Output a 1 or 8-piece cap table.)

This parameter enables the output of a capacitance table with multiple steps. Liberate characterizes and saves the capacitance data into the ldb. It can then be written into a library. The three capacitance steps are fixed at the levels specified by the control parameters `measure_slew_lower_fall`, `measure_slew_upper_fall`, and `delay_inp_fall` for a falling input transition, and by `measure_slew_lower_rise`, `measure_slew_upper_rise`, and `delay_inp_rise` for a rising input transition.

`0` Outputs 1 (when `ecsm_version` = 1.0) or 2 piece cap table (when `ecsm_version` >= 2.1). This setting is provided for backward compatibility.

`1` Outputs a 3-piece cap table when `ecsm_version` >= 2.1.

Note: To output a 3-piece cap table for input pins, we also recommend setting the parameter ccs_cap_hidden_pin to 2

`2` Outputs an 8-piece cap table when `ecsm_version` >= 2.1. The 8 pieces default to the following and can be overridden by the `set_receiver_cap_thresholds` command:

```
-rise {0.1 0.3 0.5 0.6 0.7 0.8 0.9
0.9999}
-fall {0.9 0.7 0.5 0.4 0.3 0.2 0.1
0.0001}
```

This parameter must be used before the `char_library` command.

ecsm_cap_style

`<capacitance | capacitance_set | off>`

Enables modeling of ECSM pin capacitance in a compact data structure known as `ecsm_capacitance_set`. This data structure is used when the ECSM pin capacitance does not depend on output load.
Default: `capacitance`

`capacitance` Outputs load dependent `ecsm_capacitance` value tables.

`capacitance_set`

Outputs compact `ecsm_capacitance_set` value tables. The load dependent capacitance data is verified. If the effect of the load is less than a specified tolerance (see ecsm_cap_load_effect_tol), then ignore the load effect; if not, keep all load dependent data.

`off` Disables modeling of `ecsm_cap` tables. This setting works only when the `write_library -ecsm_ccs` command option is used. As it is not needed to model both CCS and ECSM capacitance tables, this setting can be used to reduce the size of the library by removing redundant data.

This parameter must be specified before the `write_library` command.

ecsm_cap_use_input_transition

`<0 | 1>` Sets ECSM pin capacitance to follow the input transition.
Default: `1` (Follow the input.)

- `0` `ecsm_capacitance` tables follow the output transition.
- `1` `ecsm_capacitance` tables follow the input transition.

This parameter must be used before the `write_library` command.

ecsm_capacitance_factor

`<value>` Specifies a multiplication factor to apply to the ECSM capacitance data. Use this factor in combination with the `ecsm_capacitance_precision` to reduce the number of digits of ECSM capacitance output into the library. This factoring reduces the number of leading zeros without compromising accuracy.
Default: `10000`

This parameter must be specified before the `write_library` command.

ecsm_capacitance_precision

`<value>` Specifies the number of digits of precision to output into the library for ECSM capacitance data.
Default: `3`

The default value of `3` is consistent with the default tolerance for capacitance (`0.001`) in the `compare_library` utility.

Ensure that the precision is not set too small so that accuracy is not compromised.

This parameter must be specified before the `write_library` command.

ecsm_factor_mode

`<0 | 1>` Enables reduction in the physical size of the output library by factoring the ECSM data and controlling the precision. For more information, see the ecsm_capacitance_factor, ecsm_capacitance_precision, ecsm_waveform_time_factor, and `ecsm_waveform_time_precision` parameters. Default: `0`

- `0` Does not apply any factoring to the ECSM data.
- `1` Apply factoring and precision control to the ECSM data in order to reduce the physical size of the output library.

Example

The characterized ECSM capacitance data is 0.123456 ff in the `.lib`. The `ecsm_capacitance_factor` is set to `1000`. The `ecsm_capacitance_precision` is set to 5. Therefore, `0.123456 * 1000 = 123.456`; keeping five digits this data becomes 123.45 in the final library.

Note: The parameter `ldb_precision` controls the number of digits stored in the LDB.

This parameter must be specified before the `write_library` command.

ecsm_invert_gnd_current

`<0 | 1>` Inverts current values for `ecsm_current_waveform` groups. Default: `1` (Invert current.)

- `0` Does not invert currents.
- `1` Inverts the current values (`index_1`) for `ecsm_current_waveform` groups associated with ground supplies. (Currents are normally positive instead of normally negative.) This is for compatibility with EPS.

This parameter must be specified before the `char_library` command.

ecsm_measure_output_range

`<0 | 1>` Enable scaling of the voltage span for the ECSM format. The ECSM format requires that the data for the output voltage swings from Vss to Vdd. When there is an output that does not swing from Vss to Vdd, the data must be shifted such that it does swing from rail to rail (for example, Vss to Vdd).
Default: `1`

- `0` Provides backward compatibility to the 3.2 and prior releases.
- `1` Modifies the measurement algorithm to scale the voltage span according to the simulation outcome. For example, if the output swings between 0.03*Vdd and 0.94*Vdd then Liberate will sample the output crossing times at the ECSM voltage grid adjusted for the span:

  ```
  (0.03*Vdd) + (span * [0.02, 0.05, 0.1, 0.2, ... 0.9, 0.95, 0.98])
  ```

  where:

  ```
  span = (0.94*Vdd) - (0.03*Vdd)  = 0.91*Vdd
  ```

In high-leakage / low voltage threshold designs, the voltage may not start nor end at the rail. If we add a stricter voltage requirement (within `2%` of the rail instead of `5%`), then there is more characterization failures on those designs, unless this parameter is set to `1`.

In addition, the following ECSM attributes are added to the timing group:

```
ecsm_base_rail_vdd_rise
ecsm_base_rail_vdd_fall
ecsm_base_rail_gnd_rise
ecsm_base_rail_gnd_fall
```

Important

Check if your version of ETS supports these attributes before setting this parameter to `1`. If this is supported, we recommend a setting of `1`.

This parameter must be specified before the `char_library` command.

ecsm_version

`<1.1 | 2.0 | 2.1>` Specifies the ECSM format version. The supported versions are: 1.1, 2.0, and 2.1.
Default: `2.1`

`1.1`	Refers to output transition.
`2.0`	Includes information on each cell's average `N` and `P` threshold voltage, in addition to `ecsm_capacitance` information on hidden pins such as the `D` pin of a flip-flop. In ECSM 2.0 the `ecsm_capacitance` group within a `rise/fall_transition` group refers to the input pin transition not the output transition as was the case for ECSM 1.1. If no advanced multi-piece receiver capacitance data is stored in the ldb, then a single capacitance value is outputted.
`2.1`	Supports an `ecsm_capacitance` with 2-piece or 3-piece capacitance tables, dependent on the setting of ecsm_cap_mode.

This parameter can be used after the `char_library` command.

Example

```
# Set the ECSM version to 2.1
set_var ecsm_version 2.1
```

ecsm_waveform_for_bidi_pin

`<0 | 1>` Enables generation of `ecsm_waveform` groups under bidir pins.
Default: `0`

`0`	Used for backward compatibility.
`1`	Generates `ecsm_waveform` groups under bidir pins.

This parameter must be used before the `char_library` command.

ecsm_waveform_style

`<waveform | waveform_set>`

Requests output of `ecsm_waveform_set` compatible ECSM format libraries. The *ECSM Waveform Set* stores time points without including voltage because the voltages are shared. This allows for a compact representation of the ECSM model.
Default: `waveform_set`

`waveform` Uses this value for compatibility with Liberate 14.1 ISR1 and prior releases.

`waveform_set` Uses this value to enable the more compact *ECSM Waveform Set* data format.

The parameter must be specified before the `write_library` command.

ecsm_waveform_time_factor

<value> Specifies a multiplication factor to apply to the ECSM time waveform data. Use this factor in combination with the `ecsm_waveform_time_precision` to reduce the number of digits of ECSM time waveform data output into the library. This factoring reduces the number of leading zeros without compromising accuracy.
Default: `1000`

This parameter must be specified before the `write_library` command.

ecsm_waveform_time_precision

<value> Specifies the number of digits of precision to output into the library for ECSM waveform based time data.
Default: `3`

The default value of `3` is consistent with the default tolerance for capacitance (`0.001`) in the `compare_library` command.

Ensure that the precision is not set too small as this could impact accuracy.

This parameter must be specified before the `write_library` command.

ecsm_write_default_vth_to_ldb_enable

`<0 | 1>` Controls the type of threshold voltage (Vth) values that should be written into the ECSM library.
Default: `0`

- `0` Write the Vth values obtained from the simulation run into the ECSM library.
- `1` Writes the default Vth values to into the ECSM library.

This parameter must be used before the `write_library` command.

Example

```
set_var ecsm_write_default_vth_to_ldb_enable 1
```

ecsmn_loadcap_mode

`<0 | 1 | 2>` Specifies whether the miller capacitance should be included in the `ecsm_loadcap` attribute.
Default: `1`

- `0` Liberate includes wire capacitance (decoupled and cross-coupled) and miller capacitance in the `ecsm_loadcap` attribute. This setting is provided for backward compatibility with LIBERATE 14.1 ISR2.
- `1` Liberate includes wire capacitance (decoupled and cross-coupled) and device capacitance in the `ecsm_loadcap` attribute. The miller capacitance is not included.
- `2` Liberate includes wire capacitance (decoupled and cross-coupled), device capacitance, and miller capacitance in the `ecsm_loadcap` attribute.

This parameter must be used before the char_library command is run.

ecsmn_mode

`<0 | 1>` Controls ECSM Noise modeling behavior for pin-level or arc-level constructs.
Default: `1`

`0`	Use pin-based ECSM Noise models for all groups.
`1`	Generates arc-based ECSM Noise models for single-stage Channel Connected Circuits (CCC) and pin-based models for 2+ stages. This can improve accuracy at smaller geometries, and Cadence recommends it.

This parameter must be used before the char_library command is run**.**

ecsmn_skip_itt

`<0 | 1>` Controls whether to skip characterization and modeling of the ECSM internal transition tables (ITT) when characterizing an ECSM Noise model.
Default: `0`
Recommended: `1`

`0`	Allow characterization and modeling of the ECSM internal transition tables.
`1`	Skip characterization or modeling of the ECSM internal transition tables. This is the recommended value.

This parameter must be used before the char_library command is run**.**

ecsmn_vtol_mode

<0 | 1 | 2>	Requests vltolerance and vhtolerance characterization. Default: 0 Recommended: 1
0	Uses a fixed value for vltolerance and vhtolerance.
1	Computes the vltolerance and vhtolerance based on the DC transfer characteristics of the cell.
2	Disables the output of the ecsm_vltolerance and ecsm_vhtolerance attributes into the .lib.

This parameter must be used before the char_library command is run.

em_calculation_include_input

<0 | 1>	Specifies whether EM calculation should be performed on the input pins. Default: 1
0	Do not consider input pins for EM calculation.
1	Consider input pins for EM calculation.

This parameter must be used before the char_library command is run.

em_calculation_monitor_rails

<0 | 1>	Specifies if the EM calculation should be performed on the power supply rails. Runtime can be improved by skipping the rails. Default: 1
0	Limits the EM calculation to just the signal nets eliminating both supply and ground rails from being considered.

`1`	EM calculation is performed on all nets including the rails.

For more information, see the Liberate Details section on Electromigration.

This parameter must be used before the char_library command is run.

em_calculation_monitor_rails_skip_layer

<layer>	Specifies the names of specific process layers to be skipped when performing Electromigration calculation on power supply rails. Default: all layers are considered

Example

Will consider all layers except `M0` and `M1` layer for monitoring em on rail nets.
This setting affects power nets only.
All layers including `M0` and `M1` are considered for signal nets.

```
set_var em_calculation_monitor_rails 1
set_var em_calculation_monitor_rails_skip_layer {M0 M1}
```

This parameter must be set prior to the char_library command is run.

em_char_arcs_mode

`<0 | 1 | 2 | 3>`	Controls the arcs simulated during EM characterization. For EM characterization, Liberate can gather data from switching and hidden arcs in either rising or falling directions. Default: `3`
`0`	Use all switching and hidden arcs on both rising and falling directions. This option has the longest runtime and provides the most comprehensive set of EM data.

1	Same as setting 0, but use pulses with an initial rising edge (RF) and skip pulses starting with initial falling edge (FR). This setting takes approximately 50% of the runtime of option 0, and provides similar accuracy.
2	Same as setting 0, but skip hidden vectors for all related pins on switching vectors. Use hidden vectors for any input pin that is not a related pin on switching arcs. This provides the same EM data for output pins, but removes redundant or unnecessary data and runtime on input pins.
3	Combination of 1 and 2. This setting provides all necessary EM data with the best performance.

This parameter must be used before the char_library command is run.

em_clock_freq

<value>	Specifies the input pin toggle rate at which electromigration should be characterized (See char_library -em). The toggle rate is the number of transitions (including both rise and fall) on the primary input pin in one second. The value specified must be a number in Hertz. Scientific notation is supported. Default: 1e9 (1GHz)

This parameter essentially defines the simulation duration between transitions. For example, consider an em_clock_freq of 1e6. An input pin will switch once every 1000ns. Therefore, a pulse will last for 2000ns. For a combinational cell input, there will be two input (beginning at 0 and 1000ns) and two output transitions. For a sequential cell such as a flip-flop, an output pulse might require two rising clock edges. Consequently, four input clock edges (4000ns) are needed to trigger two output edges. The simulation duration would be 4000ns with four clock transitions (at 0, 1000, 2000, and 3000) with the D input typically switching at approximately 1500ns and the output transition will begin at around 0 and 2000 (responding to clock edges at 0 and 2000). To characterize a DFF for EM with a clock frequency of 1GHz, this parameter should be set to 2e9 or 2GHz.

Note: This parameter is effective only when em_freq_mode is set to 0.

For more information, see the Electromigration Models section in Chapter 7, "Performing Characterization using Liberate."

This parameter must be used before the char_library command is run.

Example

```
# Set the EM Clock Frequency to 20MHz
set_var em_clock_freq  40e6 ;# number of rises and falls (toggle rate)
```

em_current_type

`<0 | 1>`

Specifies the EM current type used for modeling. Liberate can model EM using only average currents, or the three current types: peak, avg, and rms.
Default: `1`

The parameter affects the electromigration characterization flow, which is enabled by using the `-em` option of the `char_library` command.

`0`	Model EM groups using only average currents.
`1`	Model EM groups using all 3 current types: peak, avg, and rms.

This parameter must be used before the char_library command is run.

em_data_file

<string>

Specifies the Spectre APS EMIR data file needed to characterize electromigration. If the `em_tech_file` is also provided, the `em_data_file` is ignored
Default: `""`

For more information, see the Electromigration Models section in Chapter 7, "Performing Characterization using Liberate."

This parameter must be used before the char_library command is run.

Example

```
set_var em_data_file "/home/work/MyTech.dat"
```

em_disable_switch_cell_ccsn

`<0 | 1>`

Controls whether CCSN groups should be generated for switch cells during characterization.
Default: `1`

The parameter affects the electromigration (EM) characterization flow, which is enabled by using the `-em` option of the `char_library` command.

- `0` — Enables generation of CCSN groups for switch cells. This value is given for backward compatibility with releases prior to LIBERATE 19.21.591_4 and LIBERATE 19.12.s072.
- `1` — Disables generation of CCSN groups for switch cells. Though characterization of the CCSN groups for switch cells is useful in the nominal flow, it is not recommended in the EM flow.

This parameter must be used before the char_library command is run.

em_dt_mode

`<0 | 1>`

Selects the calculation mode to be used for delta temperature values taken from the Voltus-Fi-generated `mos-region.txt` file in the electromigration (EM) thermal resistance characterization flow.
Default: `0`

Note: This parameter affects the EM thermal resistance characterization flow that is enabled using the `-em` option of the `char_library` command.

`0` Select the worst delta temperature value from the `mos-region.txt` file.

`1` Calculate the root-mean-square (RMS) of the delta temperature values from the `mos-region.txt` file.

The `em_dt_mode` parameter must be used before the char_library command is run.

During a Liberate characterization run, Voltus-Fi is called to do the EM thermal resistance analysis. In the process, Voltus-Fi saves the analysis results to the `mos-region.txt` file in the `tmp` directory, which can be specified using the tmpdir parameter. This file is then used to calculate the EM thermal resistance data based on the calculation mode identified using the `em_dt_mode` parameter and write the characterized data to the library.

If you want to save the `mos-region.txt` file for later use, use the following command:

```
set_var extsim_deck_dir "deck"
```

Then, the `mos-region.txt` file is saved to the `deck/sim.raw/` directory.

em_freq_mode

`<0 | 1>` Controls the selection of time window used for electromigration characterization.
Default: `1`

`0`	Uses the electromigration time window calculated based on frequency specified by `em_clock_freq`.
`1`	Estimates time window based on the time needed for output toggling.

This parameter must be used before the char_library command is run.

em_iacpeak_mode

`<0 | 1>` Enables to save iacpeak type from `aps`.
Default: `1`

`0`	Do not save iacpeak type information
`1`	Saves the iacpeak type from Spectre APS during EM characterization under max data. This uses the iacpeak results from APS instead of the max results.

This parameter must be specified the char_library command is run.

em_include_string

<string> Specifies the string to be used when loading the subckt netlists for EM analysis into SPECTRE.
Default and recommended: `".dspf_include"`

In MMSIM 14.1 and later versions, support for the `.inc` statement in the SPICE deck changed for EM analysis. When a `.inc` statement is used, only the top-level nodes of a testbench are analyzed for EMIR. The analysis nodes defined in the EM config file are completely ignored. Therefore, the signal nets are no longer included in the EM analysis. In order to access these nodes, the `.inc` statement needs to be changed to `.dspf_include`. Once this is done, the lower level nets appear in the EM report.

Note: ".dspf_include" is default and recommended for advanced FinFET nodes.
".inc" is recommended for 28nm nodes and older nodes.

This parameter must be specified before the char_library command is run.

Example

```
set_var em_include_string ".dspf_include"
```

em_maxcap

<0 | 1 | 2>	Specifies what type of EM data to output into the library. Default: 0
0	Outputs em_maxtoggle_rate format data.
1	Converts em_maxtoggle_rate data into maxcap table based on frequency and slew. Output EM maxcap table data.
2	Converts em_maxtoggle_rate data as in 1. Output both em_maxtoggle_rate and maxcap data.

The maxcap table values must be zero or greater and limited by the max_capacitance pin attribute. There is no pin-based EM maxcap data because this is only based on slew and not on load. For more information, see the Electromigration Models section in Chapter 7, "Performing Characterization using Liberate."

This parameter must be used before the char_library command is run.

em_maxcap_type

<avg | rms | peak>	Specifies the type of EM data to use when creating maxcap tables Default: avg

This parameter must be used before the char_library command is run.

em_maxcap_frequency

<value>

Specifies a list of frequency values to be used as `index_1` for all EM maxcap tables. The frequency unit is in Hertz, which is then normalized to the library units. If the library unit is `1ns`, the values should be in GHz. If the library unit is `100ps`, the values should be in 10GHz.

If not specified, the `index_1` frequency is chosen from the middle slew in the `em_maxtoggle_rate` table.
Default: `""`

This parameter must be specified before the char_library command is run.

em_report_data_usage_mode

`<0 | 1>`

Controls how EM violations are calculated.
Default (and recommended): `1`

- `0` — Use pass or fail rate in the EM report to calculate EM violation.
- `1` — Use "I limit" and "Current" in the EM report to calculate EM violation. Use of this setting is recommended to avoid inaccurate calculation of EM data.

EM violations can be calculated by Pass/Fail rate or "I limit" and "Current" in the EM report. Pass/fail rate has only one significant digit. When this parameter is set to `1`, Liberate can use "I limit" and "Current" that have more significant digits in the EM report to calculate EM violation.

This parameter must be used before the char_library command is run.

em_tech_file

<string>

Specifies the QRC technology file to be used to characterize electromigration. It is recommended to use a QRC technology file specified using this parameter instead of the Spectre APS EMIR data file specified using `em_data_file`.
Default: `""`

This parameter requires that the electromigration option is enabled (see char_library `-em`). For more information, see the Electromigration Models section in Chapter 7, "Performing Characterization using Liberate."

This parameter must be used before the char_library command is run.

Example

```
set_var em_tech_file "/home/work/MyTech.qrc"
```

em_trf_mode

`<0 | 1>`	Controls enabling and disabling of the EM thermal resistance characterization flow. Default: `0`
`0`	Disable EM thermal resistance characterization.
`1`	Enable EM thermal resistance characterization.

This parameter must be used before the char_library command is run.

If you have a library in which thermal resistance data has been defined at cell-level in a format such as shown in the sample below, you can choose to enable EM thermal resistance characterization by setting `em_trf_mode` to `1`:

```
cell (cell_name) {
    electromigration () {
        thermal_resistance (em_template_8x8) {
            index_1 ("...");
            index_2 ("...");
            values (...);
        }
    }
    pin (pin_name) {
        ...
    }
}
```

em_user_defined_arc_failed_msg_mode

`<0 | 1>` Controls which type of message should be printed when you define an invalid hidden power arc and whether Liberate should characterize this invalid arc during EM characterization.
Default: `1`

- `0` Ensures that a warning message is printed and characterization of the invalid arc continues.

  Note: This setting is provided for backward compatibility.
- `1` Ensures that an error message is printed while the invalid arc is not characterized and the cell related to it is marked as failed. The characterization of valid arcs continues.

This parameter requires that the electromigration option is enabled (see char_library `-em`). For more information, see the Electromigration Models section in Chapter 7, "Performing Characterization using Liberate."

This parameter must be used before the char_library command is run.

em_user_string

`<string>` Specifies a string containing Spectre-APS EMIR configuration commands that are included in the EMIR configuration file.
Default: `""`

This parameter requires that the electromigration option is enabled (see char_library `-em`). For more information, see the Electromigration Models section in Chapter 7, "Performing Characterization using Liberate."

This parameter must be used before the char_library command is run.

Example

```
set_var em_data_file "emirutil blech_length=longest_path"
```

em_user_string_append

<string> — Specifies a string that is appended to the string specified using the em_user_string parameter.
Default: ""

This parameter requires that the electromigration option is enabled (see char_library -em). For more information, see the Electromigration Models section in Chapter 7, "Performing Characterization using Liberate."

This parameter must be used before the char_library command is run.

Example

```
set_var em_user_string "emirutil Tj=..."
set_var em_user_string_append "emirutil EMOnlyICTFile=..."
```

Then, the `emir.conf` file will have the following content:

```
emirutil Tj=…
emirutil EMOnlyICTFile=...
```

em_vector_gen_mode

<0 | 1> — Enables the algorithm that determines a vector sequence required for complex cells, such as, synchronizer cells, where multiple clocks are required for proper cell initialization.
Default: 0

Note: This parameter can be set on a per-cell and per-arc basis.

- 0 — Do not analyze the cell for multiple clock initialization.
- 1 — Analyze the circuit to determine if multiple clocks are required to initialize the circuit.

This parameter requires that the electromigration option is enabled (see char_library -em). For more information, see the Electromigration Models section in Chapter 7, "Performing Characterization using Liberate."

This parameter must be used before the char_library command is run.

em_window_estimate_mode

`<0 | 1 | 2>` Enables EM window estimation on both the rising and falling edges.
Default: `2`

- `0` Enables backward compatibility to LIBERATE 14.1 ISR4.
- `1` Enables EM window estimation based on both the rising and falling edges. Setting up EM windows based on `clq->q` delays may still have constraint violations on some of the fast slew or small load data points. Liberate buffers all such windows to avoid constraint violations. To buffer only a specified set of cells that are known to have failed before, use the em_buffer_cell_pin_bounds command.
- `2` Estimates the em windows needed for the pulse generation based on the rise and fall transitions for any specific slew load. This has a visible impact if the rise and fall delays are different for a cell.

This parameter must be specified before the `char_library` command.

em_worst_rpt

`<0 | 1>` Controls generation of the EM worst report. The report contains the information about cell, pin, related_pin, when, rule, slew, load, toggle_rate, net, layer, resistor, and deck of the `em_max_toggle_rate`.
Default: `0`

Note: This parameter is .

- `0` Disables EM worst report generation.
- `1` Enables EM worst report generation.

This parameter for debug utility requires that the electromigration option is enabled (see char_library `-em`). For more information, see the Electromigration Models section in Chapter 7, “Performing Characterization using Liberate.”

This parameter must be used before the char_library command is run.

em_worst_rpt_name

<string>	Specifies the name of the em worst report. Default: `em_worst.rpt`

This parameter requires that the electromigration option is enabled (see char_library `-em`). For more information, see the Electromigration Models section in Chapter 7, "Performing Characterization using Liberate."

This parameter must be used before the char_library command is run.

Example

```
set_var em_worst_rpt 1
set_var em_worst_rpt_name "example"; # The name of the EM worst report will be "example_worst.rpt"
```

Assume that the current `.lib` is as following:

```
cell (DFFQX1) {
    pin (D) {
        electromigration () {
            related_pin : "CK";
            em_max_toggle_rate (em_template_2x0) {
                current_type : "average";
                index_1 ("0.006, 0.21");
                values ( \
                    "4714.95, 478.986" \
                );
            }
            em_max_toggle_rate (em_template_2x0) {
                current_type : "peak";
                index_1 ("0.006, 0.21");
                values ( \
                    "395948, 167588" \
                );
            }
            em_max_toggle_rate (em_template_2x0) {
                current_type : "rms";
                index_1 ("0.006, 0.21");
```

```
                values ( \
                    "294781, 181216" \
                );
            }
        }
    }
}
```

Then, `example_worst.rpt` will be generated with the following format:

```
cell, pin, related_pin, when, rule, slew, load, toggle_rate, net, layer, resistor,
deck
DFFQX1, D, , !CK * !Q, avg, 0.006,  , 4714.95, VSS, metal1_conn, rk3,
em_hidden_5_001_iter004
DFFQX1, D, , !CK * !Q, avg, 0.21,  , 478.986, VSS, metal1_conn, rk6,
em_hidden_5_001_iter004
DFFQX1, D, , !CK * Q, peak, 0.006,  , 395948, D, metal1_conn, rk62,
em_hidden_4_001_iter004
DFFQX1, D, , !CK * Q, peak, 0.21,  , 167588, VSS, metal1_conn, rk6,
em_hidden_4_001_iter004
DFFQX1, D, , !CK * Q, rms, 0.006,  , 294781, D, metal1_conn, rk62,
em_hidden_4_001_iter004
DFFQX1, D, , !CK * Q, rms, 0.21,  , 181216, VSS, metal1_conn, rk7,
em_hidden_4_001_iter004
```

enable_advance_licensed_features

`<string>`

Checks out the license on the server corresponding to the specified feature names and enables running of these features in parallel on the clients. If this parameter is not set, the client will report an error that the features are not enabled to run in parallel and exit. Default: `""`

Valid string values are "`ldbx`", "`interpolation`" and, "`ldbx interpolation`".

Example

Following is a sample `run.tcl` script to create and parallelize tasks to run LDBX on cell-level libraries.

```
set rootdir [pwd]

# setup bolt
set_var packet_arc_job_manager bolt
```

```
set_var packet_clients 5
set_var rsh_cmd "bsub -q liberate -R \"(OSREL==EE50||OSREL==EE60) rusage\[mem=1\]
span\[hosts=1\]\" -o %B/%L -e %B/%L"
set_var packet_client_health_checks 1

# enable feature that is going to run on clients
set_var enable_advance_licensed_features "ldbx"

source ${rootdir}/cells.tcl

# create tasks to run in parallel
set pvts {noms}
foreach pvt $pvts {
    foreach cell $cells {
       create_task -script $rootdir/run_ldbx.tcl -args [list $rootdir $pvt $cell]
-logdir $pvt/$cell
    }
}
# submit all tasks just created from create_task and runs them in parallel on
clients
parallelize_tasks -workdir "./clientlogs"
```

enable_command_history

`<0 | 1>` Specifies whether to maintain a log of the commands run in the current session.
Default: `0`

`0`	Do not maintain a log of the commands.
`1`	Enables logging of the commands.

enable_network_and_health_checks

`<0 | 1>` Enables network and heath checks for the bolt job distribution system. Default: `0`

Note: This parameter is available only for the bolt job distribution system.

extsim_ccs_option

`<"options">` Specifies the list of options to be used by the external SPICE simulator when characterizing CCS timing.
Default: `set to extsim_option`

This parameter only applies when the `-ccs` option is used with `char_library`. For accurate CCS current measurements it is advisable to use an accurate simulator setting. As delay and power characterization share the same simulation as CCS timing, this option also impacts NLDM delay, transition, power, and ECSM waveform values. The `options` string is passed as an `.option` line in the SPICE decks Liberate creates for characterization.

This parameter must be used before the `char_library` command.

extsim_ccsn_dc_option

`<"string">` Specifies the simulator options for CCSN DC table simulation.

This parameter must be used before the `char_library` command.

extsim_ccsn_dc_option_mode

`<0 | 1>` Controls whether the other CCSN simulation parameters should run when the extsim_ccsn_dc_option parameter is run. Default: `0`

- `0` Ensures that the extsim_ccsn_dc_option parameter will work along with the other CCSN simulation parameters, such as the extsim_ccs_option (non-SKI only), extsim_option (SKI only), and extsim_ccsn_option (both of SKI and non-SKI)
- `1` Ensures that only the extsim_ccsn_dc_option settings will work for CCSN DC table simulation.

This parameter must be specified before the `char_library` command.

extsim_ccsn_dc_sweep_option

`<"string">` Specifies DC simulation options inside the CCSN DC Sweep block. Default is an empty string, that is, `""`.

For example, the parameter values are added only to the inner loop of the DC sweep blocks as shown below:

```
ccsn_dc_sweep sweep dev=VA2 param=dc \
    values=[ -1.1 ... 2.2 ] {
        ccsn_dc_iloop dc dev=VZN \
        param=dc  <extsim_ccsn_dc_sweep_option> \
        values=[ -1.1 2.2 ]
    }
```

This parameter must be used before the `char_library` command.

extsim_ccsn_option

`<"options">` Specifies the list of options to be used by the external SPICE simulator when characterizing CCSN timing.
Default: set to `extsim_option`

This parameter specifies simulator options to apply to CCSN related simulation decks (see `char_library -ccsn`).

This parameter must be used before the `char_library` command.

extsim_cells_use_nodeset_for_io_pad

`<"list">` Specifies one or more cell(s) for which Liberate uses `.nodeset` instead of `.ic` on output pins in the simulation deck when the `char_library -io` option is specified. Liberate uses the `.ic` or `.nodeset` SPICE command to help the external simulator to initialize the circuit. In some cases, the external simulator might find that a circuit has difficulty reaching DC convergence with `.ic`, but `.nodeset` works. In other cases, the opposite maybe true. By default, this parameter is empty, which means that there is no change in the Liberate default behavior.

See the sim_init_condition parameter for details on the default behavior of Liberate.

Note: Using `.nodeset` instead of `.ic` does not guarantee that a simulator can always find DC a solution. Sometimes, the simulator options, the models, or the design itself is the issue.

Examples

To use `.nodeset` instead of `.ic` for `cell_A` and `cell_B` only:

```
set_var extsim_cells_use_nodeset_for_io_pad "cell_A cell_B"
```

To use `.nodeset` instead of `.ic` for all cells:

```
set_var extsim_cells_use_nodeset_for_io_pad "*"
```

This parameter must be used before the `char_library` command.

extsim_cmd

<string>	Specifies a command string to be used to call the external SPICE simulator. This parameter can be used to override the default command used by Liberate to call the external simulator. Default: see the information given below

This parameter can be used to override the default command used by Liberate to call the external simulator. The default settings are:

Important

The simulator specified with `extsim_cmd` ***must*** agree with the `-extsim` option specified with char_library.

The default setting for Spectre (that is, `extsim_cmd="spectre"`) is as given below:

```
# call Spectre
$extsim_cmd $extsim_cmd_option \
    sim.sp >& sim.lis
```

This parameter must be used before the `char_library` command.

Example

```
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 parameter can be used to override the default command options used by Liberate to call the external simulator. It is recommended that Spectre models are provided when using the Spectre simulator.

The default setting for Spectre is as given below:

```
extsim_cmd_option="+lqt 0"
```

To run Spectre in APS mode, set the `+aps` option:

```
set_var extsim_cmd_option +aps
```

This parameter must be used before the `char_library` command.

extsim_constraint_option

`"options"`	Specifies the list of options to be used by the external_simulator for constraint characterization (setup, hold, mpw). The options string is passed as a `.option` line in the SPICE decks that Liberate creates for constraint characterization.

- If `extsim_constraint_option` is not defined, the default is `extsim_option`.
- If `extsim_mpw_option` is not defined, the default is `extsim_constraint_option`.
- If neither `extsim_constraint_option` nor `extsim_mpw_option` is defined, the default for both the parameters is `extsim_option`.

This parameter must be used before the `char_library` command.

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. Default: *follows $TMPDIR/decks.$PID* A tarred and gzipped file is written for each cell named `<cellname>.tgz` in the named directory. 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`

SPICE decks are only saved when the `char_library -extsim` option is enabled.

Note: In multi-PVT mode, the decks are saved individually for each PVT in the following directory structure: `extsim_deck_dir/<PVT_Name>`

A file called `map.lst` is included for each cell which gives a mapping of the simulation for each saved spice deck. Every line in `map.lst` represents one simulation setup. SPICE decks used only to validate vectors are indicated by a keyword CHK at the end of the entry. These show failure when the vector is false. In this file, you can see:

1. Simulation status (PASS/FAIL)
2. Simulation deck directory
3. Pin, related-pin, and pin toggle directions
4. Simulation type (`combinational`, `three_state_enable`, `three_state_disable`,...)
5. Transition type (`rise/fall_transition`, `steady_state_current_low/high`,...)
6. WHEN condition and pin state vector (for example, from `-vector` of `define_arc`)
7. An optional CHK indicates this is an arc checking simulation

This parameter must be used before the `char_library` command.

Example

```
# Set the directory for storing SPICE decks
set_var extsim_deck_dir "/home/char/decks"
char_variation -extsim spectre -cells {INVX1 DFFX1}
```

extsim_deck_header

`"legal_spice_deck_commands"`

Specifies a string of legal SPICE commands. This string is written directly into the SPICE deck close to the top of the file.

Note: The commands specified using this parameter might be superseded during simulation by duplicate options specified in `extsim_option`, or `extsim_leakage_option`, or similar parameters.

The `extsim_deck_header` parameter is available so that the external simulator commands can be provided directly to the external simulator without having Liberate process or review them. This parameter is intended to be used in a `define_leafcell` (see the option `-extsim`) flow where some models are loaded into `read_spice` and some are not. That is, it is intended to be used to load models directly ino the external simulator engine

bypassing the Liberate circuit reading code. When you set the scale using `extsim_deck_header`, Liberate does not know the scale.

The preferred method to load the scale is in the model file. If `read_spice` is not reading the model file then use the command "`define_leafcell -scale <value> ...`" to tell Liberate what the scale is to apply to the netlist.

Note: With some simulators, this parameter defaults to "`\n.protect`". If you encounter any simulation issues caused by the presence of this default value, use the following command to override the default: `set_var extsim_deck_header ""`

This parameter must be used before the `char_library` command.

Example

```
set_var -type noise extsim_deck_header ".param myParam 1"
```

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.

This parameter must be used before the `char_library` 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_library -extsim` is specified.

This parameter 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` parameter and the `define_leafcell` command should first be set properly.

Note: Using the `define_leafcell` command sets `extsim_exclusive` to `1`. If the netlist cannot be processed, the `extsim_flatten_netlist` parameter might also need to be reset to `0`.

This parameter must be specified before the `char_library` command.

extsim_flatten_netlist

`<-1 | 0 | 1>` Only applies when extsim_model_include is used
Default: `-1 (disabled)`

`-1`	Flattens the cell netlist file but loads the model file directly with a `.inc` SPICE command when `extsim_model_include` is used and `define_leafcell` command exists.
`0`	Forces Liberate to use `.inc` SPICE command to load cell netlist when `define_leafcell` command exists.
1	This setting is an equivalent to `-1`, that is, it enables flattening of the cell netlist.

Note: A netlist must not be flattened in the electromigration (EM) flow because the DSPF format files contain information required by the simulator that gets filtered when the netlist is flattened. Because of this, the extsim_flatten_netlist parameter will be overridden to a value of `0`.

Example

Following examples shows how the combination of `extsim_model_include`/ `extsim_flatten_netlist`/`define_leafcell` changes the way that liberate composes SPICE simulation deck:

- Case 1:

  char.tcl

```
set_var extsim_model_include "/path/model.sp"
```

sim.sp

```
.inc '/path/cell.sp'
...
.inc '/path/model.sp'
```

- Case 2:

  char.tcl

  ```
  set_var extsim_model_include "/path/model.sp"
  define_leafcell -type nmos -pin_position {0 1 2 3} nch
  define_leafcell -type pmos -pin_position {0 1 2 3} pch
  ```

  sim.sp

  ```
  XMMP0_0 MP0:DRN MP0:GATE MP1:SRC VDD pch L=4.2e-08 W=6.4e-07
  XMMN0_0 MN0:DRN MN0:GATE MN1:SRC VSS nch L=4.0e-08 W=5.9e-07
  ...
  .inc '/path/model.sp'
  ```

- Case 3:

  char.tcl

  ```
  set_var extsim_model_include "/path/model.sp"
  define_leafcell -type nmos -pin_position {0 1 2 3} nch
  define_leafcell -type pmos -pin_position {0 1 2 3} pch
  set_var extsim_flatten_netlist 0
  ```

  sim.sp

  ```
  .inc '/path/cell.sp'
  ...
  .inc '/path/model.sp'
  ```

Note: In Case 3 above, the simulation decks for advanced noise models, such as CCSN, does not include the original netlist. Instead, a flattened netlist is always used. This is because these noise models require simulation of individual Channel Connected Regions (CCRs) on the boundary of the cells and not the complete cell.

This parameter must be specified before the `char_library` command.

extsim_immunity_option

"*options*"	Specifies the list of options to be used by external SPICE characterization for Noise Immunity Curves (NIC). Default for Spectre: `""` The `options` string is passed as a `.option` line in the SPICE deck.

This parameter must be used before the `char_library` command.

extsim_leakage_option

"*options*"	Specifies the list of options to be used by external SPICE simulator during leakage characterization . Default: *<see below>* The `options` string is passed as a `.option` line in the SPICE decks that Liberate creates for leakage characterization.

If `extsim_leakage_option` is not set and `extsim_option` is set, then the value of the `extsim_option` is applied to the leakage decks.

If `extsim_leakage_option` and `extsim_option` are not set, then the leakage simulations uses a Liberate default setting which can change at any time. For Spectre, these default options are:

```
"accurate nomod numdgt=6 measdgt=6 ingold=2 measout=0 gmindc=1e-14 gmin=1e-14
pivtol=1e-15"
```

It is recommended to set `extsim_leakage_option` explicitly to the desired value to ensure that the leakage simulations uses the desired simulator options.

This parameter must be used before the `char_library` command.

extsim_lic_keep

`<0 | 1>` Enables the HSPICE `-cc` mode of operation.
Default: `0`
Recommended: Set this parameter 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 parameter.

- `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_library -extsim` argument.

This parameter must be specified before the `char_library` command.

Example

```
set_var extsim_lic_keep 1
```

extsim_line_length_limit

<value> Limits the number of characters on a line in a SPICE deck.
Default: 0 (No limit.)

This parameter limits the number of characters on a line in spice deck in order to accommodate the difference in line-length between various simulators. If a line is too long, it will be broken into multiple lines, with the maximum character count as specified.

Setting this parameter to `0` means that there is no limit to the number of characters in a line.

This parameter must be specified before the `char_library` command.

extsim_model_include

<value>	Specifies a full path to a file that loads the SPICE models when using an external SPICE simulator. An error will result if a full path is *not* provided. (See error messages below.) Default: Uses flattened models in the external simulation input decks When this parameter is used, Liberate uses the file specified instead of the flattened models in the external simulation input deck. Liberate Variety places a statement such as shown below in the SPICE decks of the external simulator: `.include <extsim_model_include_file>`

In arc-based flow, use of the `extsim_model_include` parameter is supported as shown below:

```
% set_var -type <arc_type> extsim_model_include <arc_based_model_file>
```

However, ensure that a global `extsim_model_include` setting is added in the Tcl file before the `set_var -type <arc_type> extsim_model_include` command.

This means that the usage pattern should be strictly like illustrated below:

```
set_var extsim_model_include <global_model_file>
set_var -type <arc_type> extsim_model_include <arc_based_model_file>
```

This parameter must be used before the `char_library` command.

Example

```
set_var extsim_model_include "/home/user1/models/include_ff"
    ...
char_library -extsim spectre
```

Where `include_ff` contains:

```
.include '/home/user1/models/models.l' ff
```

Error Messages

- File does not exist

  ```
  *Error* : The extsim_model_include file <filename> cannot be found. Check the
  parameter for errors and rerun. Liberate will now exit.
  ```

- File is not readable

```
*Error* : The extsim_model_include file <filename> could not be read. Check the
parameter for errors and rerun. Liberate will now exit.
```

- extsim_model_include file is not a full path

```
*Warning* (set_var) : The extsim_model_include value should include a full
path. If the external simulator cannot locate the model file, it will terminate
and will cause Liberate to issue errors about failed simulations. Add the full
path and rerun.
```

extsim_model_include_mode

`<0 | 1 | 2>` Instructs the tool to add a `1e-5 Ohm` resistor to all internal cell nodes.
Default and 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 with DC convergence, Liberate adds a 1e-5 Ohm (10 micro-ohm) resistor to internal nodes of a cell. With some versions of spice, this can cause DC convergence issues with leakage simulations.

Note: When using Spectre Kernel Interface (SKI) with `extsim_model_include` and extsim_flatten_netlist=0, if extsim_model_include_mode is not 2, Liberate issues a warning and automatically set this parameter to 2.

This parameter must be used before the `char_library` command.

extsim_monitor_deck_dir

`<"directory_name">` Create a directory to store the SPICE decks affected by the extsim_monitor script.
Default: follows ${extsim_deck_dir}_monitor

This parameter specifies a directory to write all the SPICE decks affected by the `extsim_monitor` script when using an external simulator. The directory must be visible to all server and client machines. A `sim.README` file is created in each simulation directory to help trouble-shoot why this simulation was affected. See also: extsim_monitor_enable and extsim_monitor_timeout

This parameter must be used before the `char_library` command.

Example

```
# Set the directory for storing extsim monitor SPICE decks
set_var extsim_monitor_deck_dir "/home/char/decks_monitor"
```

extsim_monitor_enable

`<0 | 1 | 2>` Enables an external simulator monitoring script that can detect if the SPICE simulation is hanging, and will take corrective action. If necessary, Liberate re-submits a simulation up to "`retry_count`" times before skipping the job. Since LIBERATE 15.1 ISR3, Spectre is one of the supported external simulators.
Default and recommended: `0` (disabled)

Note: All known Spectre hangs are due to NFS issues and not directly related to any specific Spectre issue.

`0` Does not allow client external simulation process monitoring.

1	Enables an external script located in the `$ALTOSHOME/bin` to monitor the client jobs that hang while waiting for SKI or Spectre simulations to complete. *Important* The "`extsim_monitor.sh`" script is in the `$ALTOSHOME/bin` directory, but Cadence does not recommend that users modify this script.
2	Enables an internal algorithm to monitor the client jobs that hang while waiting for SKI or Spectre simulations to complete.

Note: The `sim.lis` simulator output file is monitored for updates. If the `sim.lis` file is not updated within a specified timeout (see extsim_monitor_timeout), the SKI or Spectre simulation is restarted.

See also: extsim_monitor_deck_dir and extsim_monitor_timeout

This parameter must be used before the `char_library` command.

extsim_monitor_timeout

<time>	Specifies the time in seconds that an extsim process is inactive before action is taken by the `extsim_monitor.sh` script. Default: `1800` (30 minutes)

This parameter is used to enable recovery from hanging external SPICE processes during characterization. It controls how long the `extsim_monitor.sh` script should wait for an update from the simulator prior to taking action. If the external simulator process is terminated, then Liberate will retry this simulation "`retry_count`" times before skipping this simulation. At the end of the characterization run any cells that are not fully characterized will be reported.

Any affected simulation decks will be copied to `extsim_monitor_deck_dir`. The `exstim_monitor_enable` parameter must be set in order for this parameter to have any effect.

The default value is 1800 (30 minutes). For most standard-cell library applications where single simulations complete in less than 5 minutes, the recommended value is 300 (5 minutes). For complicated or large cells with long simulation times, it may be necessary to increase this value to 3600 or larger.

See also: extsim_monitor_enable and extsim_monitor_deck_dir

This parameter must be used before the `char_library` command.

Example

```
# Set the extsim monitor timeout to 5 minutes
set_var extsim_monitor_timeout 300
```

extsim_mpw_option

<"*options*">	Specifies the options to be used for MPW characterization with external SPICE simulator.

This parameter specifies the list of options to be used by the external_simulator characterization for mpw. The options string is passed as a `.option` line in the SPICE decks that Liberate creates for constraint characterization.

- If extsim_constraint_option is not defined, the default is extsim_option.
- If extsim_mpw_option is not defined, the default is extsim_constraint_option.
- If neither extsim_constraint_option nor extsim_mpw_option is defined, the default for both the parameters is extsim_option.

This parameter must be used before the `char_library` command.

extsim_node_name_prefix

<*string*>	Specifies the generated node prefix string. Default: "`altos_`"

Use this parameter to specify a string that is used as the prefix of all Liberate generated node names when simulation decks are written out using the original node names (see `extsim_use_node_name=1`).

The prefix is used to avoid conflicts between node names generated by Liberate and those in the cell netlist, which could occur if alpha-numeric node names are used.

This parameter must be used before the `char_library` command.

extsim_option

`"options"`	Specifies the list of options to be used by external SPICE characterization for delay, power, or timing constraint characterization. Default for Spectre: `"save=none"`

The `options` string is passed as a `.option` line in the external SPICE decks Liberate creates for characterization.

Note: The last `extsim_option` ***overwites*** the previous setting.

This parameter must be used before the `char_library` command.

Example

```
# Set the .options for external SPICE, leakage and CCS
set_var extsim_option "runlvl=5"
set_var extsim_leakage_option "gmindc=1e-14 pivtol=1e-15"
set_var extsim_immunity_option "runlvl=4 rmax=24"
```

extsim_option_presim

`"options"`	Options to be used for characterization with external SPICE. Default: `""` (none)

When checking a UDA (user-defined arc) with an external simulator (`char_library -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 parameter to specify simulator options to be used during the presim stage.

This parameter must be used before the `char_library` command.

Example

```
set_var extsim_option_presim "ITL1=300"
```

extsim_reuse_ic

`<0 | 1 | 2 | 3>` Reuses DC solution for a group of transient simulations when Spectre is 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 `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, then the last timepoint of the first transient would be the IC for the second transient. Therefore, this setting is not recommended.
- `3` Reuses DC solution by adding `skipdc=useprevic` option to the `.tran` statement. (Recommended, and the only supported option for Spectre-SKI.) (Default)

Important

This parameter is supported only in Spectre.

The `extsim_reuse_ic` parameter allows for the reuse of the first DC solution in a group of `.alters`, which can significantly speed up simulations for large cells. (See the Spectre manual for a full description of all Spectre options.)

This parameter must be used before the `char_library` command.

extsim_sanitize_param_name

`<0 | 1>` Enables sanitizing of node names in the output SPICE deck.
Default: 1

- `0` No sanitization (cleaning) is done. This may lead to an early termination of the simulation because of inconsistency of node names.
- `1` Liberate will sanitize the name before using it in `.param`, `.data`, etc. sections of the output netlist for simulation.

This parameter enables Liberate to sanitize a net/port name that uses characters that could be interpreted incorrectly as an equation when using an external simulator. (Characters such as: `\`, `<`, `>`, and so on.)

This parameter must be used before the `char_library` command.

extsim_save_failed

`<none | deck | all | log | all_plus_em_reports>`

Saves the SPICE decks for failing simulations.
Default: `all`

- `none`
- `deck` Saves only the input SPICE deck.
- `all` Saves both the input deck and the output listings.
- `log` Saves only the input deck and output log file from the simulator. The data is saved in the directory defined by the extsim_deck_dir parameter.
- `all_plus_em_reports` Enables saving of simulation related files for ElectroMigration (EM) related characterization. The SPICE decks, `emir.conf`, log, `sim.measure`, `sim.print`, and EM report files for the EM simulations will be saved.

This parameter is used to save SPICE decks and listings for failed simulations.

Note: Output SPICE decks are only saved when the `char_library -extsim` option is enabled.

This parameter must be used before the `char_library` command.

extsim_save_passed

`<none | deck | all | log | all_plus_em_reports>`

Saves the SPICE decks for passing simulations.
Default: `none`

`none`

`deck` Saves only the input SPICE deck.

`all` Saves both the input deck and the output listings.

`log` Saves only the input deck and output log file from the simulator.

`all_plus_em_reports`

Enables saving of simulation-related files for electromigration (EM) related characterization. The SPICE decks, `emir.conf`, log, `sim.measure`, `sim.print`, and EM report files for the EM simulations will be saved.

This parameter is used to save SPICE decks and listings for successful simulations.

As the number of simulation decks is very large, it is advisable to only use this setting when characterizing a small number of cells. The data is saved in the directory defined by the extsim_deck_dir parameter.

Note: Output SPICE decks are only saved when the `char_library -extsim` option is enabled.

This parameter must be used before the `char_library` command.

extsim_save_verify

`<0 | 1 | 2 | 3>` Generates additional SPICE decks to verify setup, hold, recovery and/or removal.
Default: `0`

`0` Disables generation of additional SPICE decks.

`1` Enables the creation of SPICE decks that can be used to verify the library characterization without actually running the characterization. In addition to the non-constraint-related SPICE decks, special SPICE decks will be output for the constraints of setup, hold, recovery or removal timing constraints. These constraint verification SPICE decks will utilize a sweep search in SPICE. The additional decks are placed in the appropriate sub-directory within the directory defined by the extsim_deck_dir parameter and have a `"_sweep.sp"` suffix.

Note: Minimum pulse width SPICE decks will not be created.

`2` Sets extsim_save_passed always to `deck`, and an additional 'verify' deck containing offset params on both data and clock signals will be generated based on the characterization's final results (setup/hold/recovery/removal values). By adjusting the included SPICE parameter offsets, the user can also regenerate the nominal (un-degraded) value as well. The operation of `extsim_save_verify=2` is equivalent to the operation of `extsim_save_verify=1` except that it uses the final constraint results instead of a brute force sweep. This mode will also generate MPW decks. It also requires that all simulations will still be performed if `extsim_save_verify` is set to `2`.

3	Writes SPICE decks that are the same as the `char_variation -monte` flow of Liberate Variety. Following is an example deck name: `DFQD2BWP35/setup_1/setup_1_001_verify` **Note:** This parameter value is supported only in unified characterization flow of Liberate Trio. For detailed information about this flow, see: ■ Chapter 9, "Using the Multi-PVT Characterization Flow of Liberate Trio." of this reference manual, and ■ Liberate Trio Licensing section in *LIBERATE Software Licensing and Configuration Guide*

When `extsim_save_verify` is set to `2` along with EM characterizations, an extra EM verification deck is generated for every EM arc. This has a spice deck, em config file, and a run script that can be executed to verify the em results. There is one deck for every data type (avg, rms, and peak) with the input toggles reflecting the toggle rates computed. Since the EM toggle rates may be theoretical some of the EM reports may require further processing and indirect verification. There will be comments before the alter blocks such as:

```
* Entry:2 fail+119.0*r203 - Direct Verification
.alter
.param i_t0=0.0000000e+00
.param i_v0=0.0000000e+00


* Entry:3 pass-49.5*r203 - Indirect Verification
.alter
.param i_t0=0.0000000e+00
.param i_v0=0.0000000e+00
```

The above comments denote which results can be directly looked up from the reports and which one is needed to be numerically computed.

Note: All peak toggle rates have to be indirectly verified because of the min duty based formulas involved in computing the toggle rates.

Output SPICE decks are only saved when the `char_library -extsim` option is enabled.

This parameter must be used before the `char_library` command.

Example

```
set_var extsim_deck_dir          ${rundir}/spice_decks
set_var extsim_save_passed       deck
set_var extsim_save_failed       all
set_var extsim_save_verify       1
```

extsim_tar_cmd

`"string"`	Enables to tar the SPICE decks. Default: `"tar zcf"`

This parameter is used to specify the command used to compress the output SPICE decks. Set this parameter to the NULL string ("") to disable compression.

This parameter must be used before the `char_library` command.

Example

```
# Disable SPICE deck compression
set_var extsim_tar_cmd ""
```

extsim_tend_estimation_mode

`<0 | 1 | 2>`	Specifies how simulation duration should be determined for external simulators (see also tran_tend_estimation_mode). Default: `0`

`0`	Estimates the end of simulation from transition of primary output if any.
`1`	Checks the transitioning internal wires for hidden arcs (if not SKI).
`2`	Checks all transitioning wires for hidden and switching arcs, including the ones for SKI.

This parameter must be used before the `char_library` command.

extsim_timestep

<value>	Sets the time step to use for external SPICE simulation. Default: `1e-12 seconds` (1ps)

This parameter must be used before the `char_library` command.

Example

```
# Set the time step for external SPICE
set_var extsim_timestep 2e-12
```

extsim_tran_append

<"options">	Adds additional options to the `.tran` statement. Default: `""` (none)

This parameter must be used before the `char_library` command.

Example

```
# Set conservative mode for Spectre
set_var extsim_tran_append "errpreset=conservative"
```

extsim_tran_append_skipdc

`<0 | 1 | 2 >`	Controls replacement of `skipdc=useprevic` by `useprevic=ns` in the transient analysis statement. Default: `1`	
	`0`	Specifies to not append `skipdc=useprevic` or `useprevic=ns` into the transient analysis statement.
	`1`	Specifies to use `skipdc=useprevic` in the transient analysis statement.
	`2`	Specifies to use `useprevic=ns` instead of `skipdc=useprevic` in the transient analysis statement.

This parameter must be used before the `char_library` command.

extsim_use_node_name

`<0 | 1>` Maps node names to numbers in extsim SPICE decks.
Default and recommended: `1`

- `0` Specifies to map the node names to numbers in extsim SPICE decks.
- `1` Specifies that if a port name has an escape character "\", Liberate should remove this character and replace it by '_' by default because SPICE simulators (HSPIC, Spectre) do not support this character in voltage/ current source commands.

This parameter must be used before the `char_library` command.

floating_channel_bias

`<value>` Specifies to bias the initial conditions for floating channel nodes in SPICE decks, where, -0.5 <= value <= 1.5.
Default: `0.5`

- `0` Sets the initial condition for floating channel nets to the expected final state for the net.
- `1` Sets the initial condition for floating channel nets to opposite rather than the expected final state.

Use this parameter to bias the initial conditions for floating channel nodes in SPICE decks when enabled by setting the `floating_channel_mode` parameter to `3`. Liberate attempts to determine the expected final state for each floating channel node. A value of `0` will bias the initial condition to an expected final state. A value of `1` will bias the initial condition to the state opposite from the expected final state. Any other value will bias the initial condition between the expected final state and its opposite value.

This parameter must be used before the `char_library` command.

floating_channel_mode

`<0 | 1 | 2 | 3>` Enables various algorithms to initialize floating channel nodes. Default: `1`

`0`	Initialize floating channel nodes if they drive the gate of a loading transistor.
`1`	Never initializes floating channel nodes and leaves it to the simulator to determine the initial condition. (Default and recommended)
`2`	Always initializes floating channel nodes to: gnd+0.5*(vdd-gnd) = 0.5*(vdd+gnd).
`3`	Always initializes floating channel nodes based on the setting of the floating_channel_bias parameter.

Liberate provides initial conditions to nodes that are on the active path. Nodes that are not on the active path are not initialized. It is up to the circuit simulator to provide a DC solution for these nodes. This parameter can be used to enable various algorithms to initialize floating channel nodes.

This parameter must be used before the `char_library` command.

floating_node_initialize_mode

`<0 | 1 | 2>` Specifies the initialization method for floating nodes. Default and recommended: `0`

`0`	Adds 1/gmin gshunt resistance on all floating nodes to a ground supply node.
`1`	Adds a `.ic` (initial condition) set to `0V` on all floating nodes.
`2`	Adds a `.NODESET` set to `0V` on all floating nodes

Liberate automatically recognizes floating nodes and adds `.ic` to these nodes to make the simulation results consistent across different simulators. A change to the initialization method may result in different simulation results.

This parameter must be used before the `char_library` command.

force_avg_default_select_order

`<0 | 1>` Enforces a pre-determined order for selecting average default groups.
Default: `1`

- `0` Liberate will use its internal ordering to decide the default group.
- `1` Enforces a deterministic ordering. (Default and Recommended.)

When the criteria for selecting the default group is set to the average, Liberate will select the group closest to the statistical median value as the default group. This parameter is needed to resolve differences in default groups for cells with exactly two states.

This parameter must be used before the `write_library` command.

force_condition

`<0 | 1 | 2 | 5>` Controls whether to include conditional `when` and `sdf_cond` attributes for single or binate timing groups.
Default: `1`

- `0` Disables the `when/sdf_cond` condition for arcs with a single or binate (an arc with a pair of `positive_unate` and `negative_unate` timing groups) timing group. With many sequential cells, the clock to Q path may contain many states. Some of these states may contain only rise or fall arcs while others contain both rise and fall arcs. Liberate models this arc as a single worst case unconditional arc.
- `1` Outputs `when` and `sdf_cond` conditions for 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).
- `2` In addition to mode `1`, always output `when`, for example, for AND gate.

3 4	**Note:** From LIBERATE 18.1 ISR2 onward, values 3 and 4 have been deprecated. For functionality similar to that supported by these values, set force_condition to 5 instead.
5	In addition to mode 1, for sequential switching arcs (CK->Q), output rise and fall groups are combined into a single timing group to minimize the pins in the combined when condition. For example, Q rise when SE*SI and Q fall with SE*!SI would combine into a single timing group with when SE. For cases where the number of rise and fall groups is not equal, the remaining groups are left unchanged. For example a FF with data input !C*D could produce a timing group with cell_rise and cell_fall when !C, and two timing groups with only cell_fall when C*D and when C*!D.

This parameter must be used before the char_library command.

Example

```
# Disable conditions on single timing groups
set_var force_condition 0
```

force_default_group

<0 | 1>	Controls whether to include a default group. Default: 0
0	Specifies that Liberate should not always output a default group.
1	Requests Liberate to always output a default group, even if the group is redundant. This can occur when all states are exhaustively enumerated.

This parameter must be used before the `char_library` command.

Example

```
# Require output of default group
set_var force_default_group 1
```

force_edge_timing_type

`<0 | 1 | 2>` Controls the `timing_sense` and `timing_type` of single-sided edge timing arcs.
Default: `1`

- `0` Convert `rising_edge` and `falling_edge` to `combinational_rise` and `combinational_fall` for single sided edge-triggered timing() groups. An appropriate timing_sense (`negative_unate` and `positive_unate`) will be output instead of `non_unate`.
- `1` Do not change the `timing_type` of `rising_edge` and `falling_edge` for single sided edge triggered arcs. If appropriate, change the timing sense to `negative_unate` and `positive_unate` from `non_unate`. This is the default and recommended setting.
- `2` Convert the `timing_type` for single sided timing arcs where the `related_pin` is a clock from `combinational_rise` and `combinational_fall` to `rising_edge` and `falling_edge`.

Use this parameter to request specific handling of single sided edge triggered timing arcs. Normally, Liberate only outputs timing() groups with an edge timing types (`rising_edge` or `falling_edge`) where both rising and falling data exist in a single group.

If the define_arc command is specified in the template, the `timing_type` is not modified.

The following parameters (there might be others) might also affect the `timing_type`/ `timing_sense` of characterized arcs: combinational_risefall, nonseq_as_recrem, merge_related_preset_clear, and discard_timing_sense_after_merge.

This parameter must be used before the `char_library` command.

force_leakage_if_no_pg_pin

`<0 | 1>` Enables output of leakage when no `related_pg_pin` is available.
Default: `1`

- `0` Restores the behavior of 3.0 and prior releases where leakage groups were not output if there was no `related_pg_pin`.
- `1` Requests Liberate to force 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.

Important

The force_leakage_if_no_pg_pin parameter only works with pin_based_leakage and it will only affect cells without related pg pins for leakage power.

To be compliant with LC, the `cell_leakage_power` for that cell is output to the library, without regard for the value of the `cell_leakage_power` parameter. The default leakage power for that cell will not be output to the library, without regard for the value of the `keep_default_leakage` parameter.

This parameter must be used before the `write_library` command.

force_related_power_pin

`<0 | 1 | 2>` Forces the output of related_power_pin groups for cells.
Default and recommended: `2`

If there is no related pg pin found in the netlist of a cell, this parameter controls whether to force to output the related power pin or signal level to the default vdd/gnd or the vdd/gnd set by `set_pin_vdd`/`set_pin_gnd`.

- `0` Do not force the output of `related_power_pin` data.

1	Forces this behavior only for cells that have more than one pin (Antenna cells are not forced).
2	Forces this behavior for every cell.

This parameter must be used before the `write_library` command.

force_timing_type

`<0 | 1>` Forces Liberate to output the `timing_type` attribute.
Default: `0`

Use this parameter when Liberate does not output the `timing_type` attribute and defaults to `combinational` because the Liberty default `timing_type` is `combinational`.

0	The `timing_type` is not required in the output library for arcs with a `combinational` timing type.
1	Forces the output of the attribute `"timing_type: combinational"` for all arcs where Liberate has no other `timing_type`.

This parameter must be used before the `write_library` command.

force_unconnected_pg_pin

`<0 | 1>` Controls whether pg_pin information will be generated for unconnected power supplies.
Default: `0` (no pg_pin for supplies with no connections)

0	pg_pin information for unconnected power supplies will *not* be output.
1	pg_pin information for unconnected power supplies will be output.

The `set_vdd` and `set_gnd` commands specify the power supplies. Normally, if a subcircuit definition for a cell is not connected to a specified supply, that supply will not be included in

the pg_pin group for that cell. Set this parameter to force all specified supplies to be output in pg_pin groups.

This parameter must be used before the `char_library` command.

group_attribute

`<"attribute_name">`	Creates a cell group based on the named attribute. Default: `""` (if not set, then the library attribute `cell_footprint` is used)

A cell group may be created explicitly by the define_group command, or implicitly via cells having a common cell level attribute as specified by this parameter. Cell grouping works with a `read_library` or `write_template` flow, `compare_library -group`, and `compare_structure` (only available in Liberate LV). Cells that have the same value for a selected attribute are grouped together (for example, "cell_footprint").

Example

```
set_var group_attribute "cell_footprint"
```

heartbeat_initial_timeout

`<time>`	The time in seconds that the server will wait for the first client to communicate back to the server. Default: `3600` (1 hour)

This parameter is used to specify a time limit to wait for a client process to communicate with the Liberate server jobs. When this timeout is exceeded, the Liberate server will issue a warning that the client has failed to start and then restart the `heartbeat_initial_timeout` timer. This situation could occur, for example, due to network problems.

This parameter must be used before the `char_library` command.

Example

```
# Set the heartbeat initial timeout to 2 hours
set_var heartbeat_initial_timeout 7200
```

heartbeat_timeout

<time>	The time in seconds that a client machine is inactive before being released by the server machine. Default: `300` (5 minutes)

This parameter is used to enable recovery from machine failures during distributed characterization. It controls how long the server machine should wait for a response from a client before releasing that client. If a client hangs it will not be used for the remainder of the characterization 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 characterization run any cells that are not fully characterized will be reported.

This parameter must be used before the `char_library` command.

Example

```
# Set the heartbeat timeout to 10 minutes
set_var heartbeat_timeout 600
```

hidden_power

`<0 | 1>`	Enables conditional hidden power calculations for all cells. Default: `1`

This parameter is used to enable or disable "hidden" power calculations for all cells, including input pins of combinatorial gates. When this parameter is disabled hidden power is only calculated for hidden input pins (that is, pins that have no path to the output such as the D pin of a flip-flop). When `hidden_power` is enabled, conditional hidden power will be calculated for all pins that have can have a logic state that does not toggle an output node - for example, a 2 input NAND gate where one of the side inputs is zero.

This parameter must be used before the `char_library` command.

Example

```
# Disable hidden power calculations
set_var hidden_power 0
```

immunity_glitch_peak

<value>	The fraction of the output supply voltage used for characterizing noise immunity curves. Default: `0.05` (5%)

This parameter is used to set the noise glitch-peak that specifies the failure criteria for the output of the cell when generating noise immunity curves. Default is 5% of the supply voltage.

This parameter must be used before the `char_library` command.

Example

```
# Set the acceptable glitch output peak to 10%
set_var immunity_glitch_peak 0.1
```

immunity_noise_skew_ratio

<value>	Ratio of time to reach noise-peak compared to the noise width. Default: `0.5`

This parameter is used to set the skew of the input noise waveform used for characterization of noise immunity rejection curves. It defines the ratio of the time to reach the noise peak over the noise width. The ratio must be within a range of 0.25 to 0.75, default is 0.5 (the input glitch is represented by an isosceles triangle).

This parameter must be used before the `char_library` command.

Example

```
# Set maximum noise peak to occur at 40% of the width
set_var immunity_noise_skew_ratio 0.4
```

init_clock_period_mode

`<0 | 1 | 2>` Specifies initialization sequence for obtaining initial node voltages for simulation.
Default: `0`

`0`	Apply `init_constraint_period` as an input pulse to the related_pin and other side pins and capture the initial node voltages. Apply these initial node voltages to subsequent simulations for this vector. (Default)
`1`	Get the initial node voltages from #**0** above and if these node voltages do not agree with the *Inside View* algorithm's prediction of the circuit behavior, then the `init_constraint_period` parameter is silently disabled for this arc.
`2`	Use the behavior as in `1` above, but apply the `init_constraint_period` only to the related_pin. No side pins will be pulsed during the initialization sequence. (Recommended)

This parameter must be used before the `char_library` command.

init_comb_num_cycles

`<integer>` Number of times the related_pin will be toggled. Default: 1

This parameter lets you specify how many times the related_pin in combinational arcs will be toggled. Allowable values are integers starting from 1.

Note: Only when the `init_comb_related_pin_period` parameter is set to a positive number will `init_comb_num_cycles` be used. Otherwise, the related_pin will not be toggled.

This parameter must be used before the `char_library` command.

Example

```
set_var init_comb_num_cycles 2
```

init_comb_related_pin_period

<integer> Controls related_pin toggling in combinational arcs.
Default: `-1`

Setting this parameter to a positive number (units in seconds) causes the related_pin in combinational arcs to be toggled. The period of the toggle is set by this parameter while the number of toggles is set by the `init_comb_num_cycles` parameter.

This parameter must be used before the `char_library` command.

Example

```
set_var init_comb_related_pin_period 10e-9
```

init_constraint_period

<value> Specifies the clock period (in seconds) used for circuit initialization during constraint characterization.
Default: `-1` (do not perform extra initialization)

A clock pulse with a 50% duty cycle will be used to initialize the circuit before constraint circuit simulation. Default: no initial clock pulse is used.

This parameter must be used before the `char_library` command.

Example

```
# Use a clock pulse with a 5ns period for initialization
set_var init_constraint_period 5e-9
```

init_constraint_period_binning_mode

`<0 | 1>` Use `init_constraint_period` for binning simulation.
Default: `1`

`0` Behavior of release 12.1.1 and earlier

`1`	Use `init_constraint_period` for binning simulation. (Default and Recommended)

This parameter must be used before the `char_library` command.

init_constraint_period_check_mode

`<0 | 1>` Controls how vector pre-pulsing is applied.
Default: `0`

`0`	Check to make sure the entire vector is identical before-and-after pre-pulsing.
`1`	Loosens the check to just the probe node.

When `init_constraint_period` is enabled, Liberate checks to make sure the entire vector is identical before and after the pre-pulsing, otherwise pre-pulsing is disabled.

For certain cells with multiple internal latch nodes, pre-pulsing should only be concerned with the state of the setup/hold probe (usually the first latch node.) This results in setup/hold differences after logic constraints are applied, even though all vectors should give similar setup/hold results. The difference can be attributed to pre-pulsing. A vector that has all 0's or all 1's have a higher likelihood of passing Liberate's pre-pulsing check.

This parameter must be used before the `char_library` command.

init_delay_period

`<value>` Clock period (in seconds) used for circuit initialization during delay characterization.
Default: `-1` (do not perform extra initialization)

This parameter is used to specify a clock period (in seconds). A clock pulse with a 50% duty cycle will be used to initialize sequential cells before delay characterization. Affects clock to output delay arcs only. Default: no initial clock pulse is used.

This parameter must be used before the `char_library` command.

Example

```
# Use a clock pulse with a 5ns period for initialization
set_var init_delay_period 5e-9
```

init_pin_hidden_period

<value>	Time period (in seconds) used for circuit initialization during hidden arc characterization. Default: 10e-9 seconds (10ns).

This parameter must be used before the `char_library` command.

init_pin_hidden_num_cycles

<integer>	Number of times the related_pin will be toggled. Default: 1

Use this parameter to specify how many times the related_pin in hidden power arcs will be toggled when using a transient initialization. This can be useful when using `init_pin_hidden_period` to initialize circuits such as synchronizer circuits that require more than one toggle. The valid values are integers starting from 1.

Note: The `init_pin_hidden_period` parameter is used only when the `init_pin_hidden_period` parameter is set to a positive number. Otherwise, the related_pin will not be toggled.

This parameter must be used before the `char_library` command.

Example

```
set_var init_pin_hidden_period 1e-9
set_var init_pin_hidden_num_cycles 2
```

init_pin_hidden_period_mode

`<0 | 1>`	Additional effort can be made to ensure that the initialization pulse puts the cell in the correct initial state. Default: 1 (Use additional effort)
	`0` Standard method.

1	Extra effort is used to make certain the cell is properly initialized. (Recommended)

If `init_pin_hidden_period_mode=1` and either `init_combinational_period>0` or `init_delay_period>0`, then additional effort is made to ensure that the initialization pulse puts the cell in the correct initial state for the measurement. The effect of using this option will be seen primarily on hidden power measurements of clock pins on sequential cells.

This parameter must be used before the `char_library` command.

input_noise

`<dc | hyper | both>` Specifies SI constructs to write for input pins.
Default: `hyper`

`dc`	Causes `input_voltage` DC noise level attributes to be written
`hyper`	Outputs noise rejection curves (`hyperbolic_noise_low`, `hyperbolic_noise_high`).
`both`	Enables both DC noise and hyperbolic constructs to be written.

This parameter is used by `write_library` to specify the SI constructs to output for input pins (including bi-directional pins). This is in addition to any noise-immunity table that may exist between an output pin and that input pin.

This parameter must be used before the `char_library` command.

Example

```
# Output DC noise for each input pin
set_var input_noise dc
```

input_output_voltage

`<0 | 1 | 2>` Creates `input_voltage` and `output_voltage` groups in the library.
Default: `0` (Do not create these groups.)

0	Do not create `input_voltage`, `output_voltage` groups.

1	Create `input_voltage`, `output_voltage` groups and use VDD/GND (rail voltages) for vih,voh and vil,vol. For example, for VDD=1.0 and GND=0.0 with 20% and 80% slew thresholds:
	`input_voltage(default_VDD_GND_input) {`
	`  vil: 0.0`
	`  vih: 1.0`
	`  vimin: 0.0`
	`  vimax: 1.0`
	`}`

2	Create `input_voltage`, `output_voltage` groups and scale values by upper and lower slew thresholds. For example, for VDD=1.0 and GND=0.0 with 20% and 80% slew thresholds:
	`input_voltage(default_VDD_GND_input) {`
	`  vil: 0.2`
	`  vih: 0.8`
	`  vimin: 0.0`
	`  vimax: 1.0`
	`}`

io_mode

`<-1 | 0 | 1>` Controls enabling and disabling of IO mode. Default: `-1`

The `io_mode` parameter takes precedence when it co-exists with the `-io` option of the char_library or `char_variation` command.

`-1`	Follow the `-io` option set with the `char_library` or `char_variation` command.
`0`	Disable IO mode even if the `-io` option is specified with the `char_library` or `char_variation` command.
`1`	Enable IO mode. It functions the same way as does the `-io` option of the `char_library` or `char_variation` command.

This parameter can be set on a per-cell basis using the `-cell` option with the set_var command.

Example

```
# characterize INVD1 in IO mode and characterize INVD2 in regular mode
set_var -cell {INVD1} io_mode 1
char_library -extsim spectre -thread 1 -cells {INVD1 INVD2}
```

keep_dcap_leakage

`<0 | 1>` Controls whether to keep the default DCAP leakage group. Default: `1` (Keep)

`0`	Do not keep the default leakage groups for decap cells. See also keep_default_leakage_group.
`1`	Keep the default DCAP leakage group.

This parameter can be used after the `char_library` command.

keep_default_leakage_group

`<0 | 2>` Controls output of default leakage power group.
Default: `0` (Disable output of default `leakage_power` groups.)

- `0` Disables the output of the default `leakage_power` groups.
- `2` Liberate will skip all leakage group processing.

The parameters keep_dcap_leakage and `keep_default_leakage_group` are related and only affect cells without pins, such as DCAP cells. They have no affect on cells with one or more pins. For cells without pins, the interaction is as following:

keep_dcap_ leakage	keep_default_ leakage_group	Result
0	0	All leakage groups removed.
0	2	Leakage groups unchanged.
1	0	`voltage_map = 0`: All leakage groups removed. `voltage_map != 0` or CCSP: Only default leakage group remains.
1	2	Leakage groups unchanged.

This parameter can be used after the `char_library` command.

keep_empty_cells

`<0 | 1>` Determines how to handle a cell when the netlist is empty.
Default: `0` (Treat empty cells as an error condition.)

- `0` Generates an error condition when an empty cell subcircuit is found. The cell will not be included in the .lib file. (Default)
- `1` Generates a warning condition when an empty cell subcircuit is found. The cell will be included in the .lib file, but will not have any data.

A cell is available for characterization when all of the following conditions are true:

- There is a `define_cell` command and a netlist that has been read in using the `read_spice` command.
- The `char_library` command does not have the `-cells` argument, or the `-cells` argument includes the cell name and the `-exclude` argument is not used.

This parameter must be used before the `char_library` command.

keep_user_defined_arc_failed_data

`<0 | 1>`	Keep user specified arcs (see `define_arc`) even if they cannot be characterized. Default: `1` (Do not remove bad data.)

When Liberate is provided with a `define_arc` for hidden power and Liberate determines that the arc is not a valid arc, this control parameter can be reset (set to 0) to remove the whole failed user defined hidden power arc from the characterized data (ldb). Once removed, if needed, Liberate will add the missing data as a `scalar 0` data matrix to make LC pass.

This parameter must be used before the `char_library` command.

ldb_checkpoint_dir

<dir>	Directory where the ldb checkpoint file will be stored. Default: "." (The initial run directory)

Liberate stores the characterization data in a temporary ldb (library database) file called *altos.ldb.<PID>* where PID is the process ID. This parameter is used to specify the directory where the temporary ldb checkpoint file will be stored. The default directory is the directory where the initial run was started.

This parameter must be used before the `char_library` command.

Example

```
set_var ldb_checkpoint_dir /home/work/rundir
```

ldb_precision

<positive_integer>

Specifies the precision used in the ldb.
Default: 6 *(See note below.)*

This specifies the precision used internally in the ldb.

Note: If this parameter is not specified, precision defaults to `6` places.

Precision can only be specified with a positive integer. Examples:

```
set_var ldb_precision 8     # Means "%.8g" or "%.8f".
set_var ldb_precision 0     # Means "%.0g" or "%.0f"
set_var ldb_precision 08    # Same as "8"
```

The following are invalid settings:

```
set_var ldb_precision abc   # Not an integer
set_var ldb_precision "2 "  # No white space
set_var ldb_precision ""    # "(nil)" is illegal
set_var ldb_precision 1e+2  # No scientific notation
set_var ldb_precision 1e-2  #  - same error -
set_var ldb_precision 8.0   # No floats (integers only)
```

A warning message will be output if invalid settings are encountered.

This parameter must be used before the `char_library` command.

ldb_save_all_cells

`<0 | 1>`

Controls if cells in the input ldb not being recharacterized are written out into the output ldb.
Default: `1`

When set to 1, the `ldb_save_all_cells` parameter ensures that cells in the input ldb that aren't being recharacterized are still written out into the output ldb. To get the pre 3.0p2 and prior behavior, set this parameter to 0. The recommended setting is 1, which is also the default.

This parameter must be used before the `char_library` command.

leakage_add_input_pin

`<0 | 1 | 2>` Include or exclude input pin leakage power.
Default: `1` (Include input pin leakage.)

- `0` Disables the inclusion of input pin leakage with reported leakage. (Set this to achieve the behavior of releases prior to 3.1, which only reported input leakage when voltage_map=0.)
- `1` Include input pin leakage. (Default.)
- `2` Account for leakage from bi-directional pins.

This parameter must be used before the `char_library` command.

leakage_add_missing_group

`<0 | 1>` Prevents Liberate from adding missing leakage groups.
Default: `1`

By default, for level shifters, when the input voltage comes from a voltage domain that is not specified as a pg_pin, setting this parameter to 0 will stop Liberate from adding the missing leakage groups, thus avoiding an LC error.

This parameter must be used before the `char_library` command.

leakage_cell_attribute

`<0 | 1 | 2>` Controls inclusion of the `cell_leakage_power` attributes in the output libraries.
Default: `1`

- `0` Specifies to omit the `cell_leakage_power` attributes from all output libraries.
- `1` Specifies to include the `cell_leakage_power` attribute in the CCSP format output libraries except when the default leakage group is included (see set_default_group).

`2`	Specifies to always include the `cell_leakage_power` attribute in the CCSP format output libraries.

This parameter can be used after the `char_library` command.

leakage_expand_state

`<0 | off | 1 | whens | 2 | vectors>`

Selects different algorithms for expanding leakage states. Default: `0 | off`.

`0 | off`	Select one vector per when condition, or per define_leakage command, if specified.
`1 | whens`	Characterize all vectors that match the specified "when". The when is adjusted to include all pins included in the vector. The `define_leakage` command must be used to specify the desired "when" leakage states.
`2 | vectors`	Characterize all vectors that match the specified "when". The original "when" leakage states will be maintained and the combination (worst, best, or average) of the vectors will be modeled according to the criteria specified in the set_default_group command. If mega-mode is used (see `mega_enable` or the `-type` option of `define_cell`), all bundle or bus pins will be removed from the specified "when" expressions and the characterized vectors will be limited to the bundle combinations specified (see `mega_bundle_mode`, `define_bundle_pins`, and `define_bus`).

This parameter must be used before the `char_library` command.

leakage_float_internal_supply

`<0 | 1 | 2>` Affects leakage measurements for power switch cells.
Default: `1`

Note: Set this parameter for increased pessimism of leakage characterization for power switch cells.

`0` The following behavior applies for all header and footer cell internal supply pins that are specified using define_cell `-internal_supply`:

- If the power switch is on, leave the `internal_supply` pin floating.
- If the power switch is off, connect the `internal_supply` pin to a created voltage source connected to the opposite supply voltage. The current through this created voltage source will not be monitored.

  Note: For header (footer) cells, the opposite state would be ground (vdd).

`1` The `internal_supply` pin will always float when measuring leakage.

`2` The behavior will match the setting of 0 with the exception that the current through the created voltage source will be monitored and for a header (footer) cell, the current will be added to the related ground (vdd) pin. The user must specify the related ground (vdd) pin using the set_pin_gnd (set_pin_vdd) command.

This parameter must be used before the `char_library` command.

Example

```
define_cell \
  -input { sleep } \
  -internal_supply  { vdda } \
  -delay delay_template_7x7 \
  -power power_template_7x7 \
```

```
  header

define_cell \
  -input { sleepn } \
  -internal_supply  { vssa } \
  -delay delay_template_7x7 \
  -power power_template_7x7 \
  footer

set_pin_gnd -supply_name vss header vdda $vss
set_pin_vdd -supply_name vdd footer vssa $vdd
```

leakage_force_tristate_pin

`<0 | 1 | 2 | 3>` Controls how to force tristate pins output *cell_leakage_power*. This parameter can be used to match legacy libraries.
Default: `0`

`0`	Allows the tristate pins to be floating output.
`1`	Forces a tristate pin to ground during the leakage simulations.
`2`	Ties floating output and bidi pins to a voltage source at logical `0` or `1`.
`3`	Ties floating output and bidi pins to the appropriate VDD or GND supply pins to produce logical `0` or `1`. This includes the current passing through the floating pin in the leakage calculation.

This parameter must be used before the `char_library` command.

leakage_merge_state

`<0 | 1 | 2>` Specifies the leakage merge algorithm.
Default: `0`

`0`	Outputs a single leakage group for each unique state. The value is selected based on the `-criteria { leakage ...}` setting of `set_default_group`. With this setting, if the toggle_leakage_state parameter is set to 1, the output pins in sequential cells are dropped from `'when'`.
`1`	Outputs a single leakage group for each unique state. The value is the minimum leakage from the characterized values.
`2`	Outputs a single leakage group for each unique state. The value is the maximum leakage from the characterized values.

Use this parameter to tell Liberate how to select the leakage value to report when there are multiple leakage measurements with the same '`when`' state. The output library should have only one leakage for each state. Merging of leakage states might be needed, For example, when there are user-specified `define_leakage` commands whose states do not include all of the cell pins.

This parameter must be set before the `char_library` command.

leakage_mode

`<0 | 1 | 2>`	Set the leakage computation mode. Default: `1` (multiply the leakage current by the supply voltage)

This parameter specifies the method used when computing the DC leakage that needs to be reported in the .lib file. The reported leakage is computed as follows:

If `voltage_map=0`, leakage is reported *without* `related_pg_pin` as follows:

leakage_mode	Formula
0	$(-I_{VDD} \times V_{SWING}) - (I_{INPUT\_HIGH} \times V_{INPUT\_SWING})$

1	$(-I_{VDD} \times V_{VDD}) + (I_{VSS} \times V_{VSS}) - (I_{INPUT\_HIGH} \times V_{INPUT\_HIGH}) + (I_{INPUT\_LOW} \times V_{INPUT\_LOW})$ If $V_{VSS} = V_{INPUT\_LOW} = 0$ then the above formula reduces to: $(-I_{VDD} \times V_{VDD}) - (I_{INPUT\_HIGH} \times V_{INPUT\_HIGH})$
2	$(-I_{VSS} \times V_{SWING}) - (I_{INPUT\_LOW} \times V_{INPUT\_SWING})$

If `voltage_map=1`, leakage is reported *with* `related_pg_pin` as follows:

leakage_mode	related_pg_pin	Formula
0 or 2	VDD VSS	$(-I_{VDD} \times V_{SWING}) - (I_{INPUT\_HIGH} \times V_{INPUT\_SWING})$ $(I_{VSS} \times V_{SWING}) + (I_{INPUT\_LOW} \times V_{INPUT\_SWING})$
1	VDD VSS	$(-I_{VDD} \times V_{VDD}) - (I_{INPUT\_HIGH} \times V_{INPUT\_HIGH})$ $(I_{VSS} \times V_{VSS}) + (I_{INPUT\_LOW} \times V_{INPUT\_LOW})$

Note: All current in the formulas above are the actual measured currents. If multiple supply rails (like VDD, VDDL) are associated with a single ground rail (VSS), the power rail selection is random. The I_{INPUT_HIGH} and I_{INPUT_LOW} are the current measured at the input pin when the input is in a logic High or Low State. The V_{INPUT_HIGH} and V_{INPUT_LOW} are the input voltages. The above formulas assume that leakage_add_input_pin is set to `1`. When this parameter is set to `0`, then all input leakage components in above formula are ignored.

This parameter must be used before the `char_library` command.

leakage_model_internal_pin

`<0 | 1>`	Controls if internal pins are used in leakage "when" state. Default: `1`
`0`	Liberate removes the internal pins from being modeled. The `internal_pin` group and the `internal_pin` in the leakage when condition are not output.
`1`	Liberate uses internal pins in leakage `when` state.

This parameter must be used before the `char_library` command.

leakage_precision

<value>	Specify leakage precision (Default: "%g")

Set this parameter to the desired precision using standard "C" print formats for the values under the *leakage_power*, *gate_leakage* and *pg_current* groups. If not set, leakage_power defaults to using **"%g"** and gate_leakage to the `write_library -precision` value.

This parameter can be used after the `char_library` command.

leakage_ramp_vsrc

`<0 | 1>` Controls whether voltage sources should be initialized to 0V at time zero and ramped to their intended value at the time equal to half of `leakage_sim_duration`.
Default: `0`

`0`	Does not apply the functionality.
`1`	Liberate initializes all voltage sources to 0V at time zero, and then ramps those voltage sources to their intended value at a time equal to half of the `leakage_sim_duration` parameter. If `leakage_sim_duration` is not set or if it is set to a value of less than 1ns, the `leakage_sim_duration` parameter is automatically forced to 1ns.

This parameter must be used before the `char_library` command.

leakage_sim_duration

<value> If non-zero, use transient simulation after the specified delay for leakage. Default: 0 seconds (use dc solution)

Use this parameter to enable a transient simulation and to specify the simulation duration to be used when measuring leakage. Setting it to 2e-12(seconds) will run the transient simulation from 0s to 2e-12s and measure the leakage at the time (value) specified minus 1ps. This parameter works with Alspice and with external simulators.

This parameter also affects the rise time when ramping the power supplies. See leakage_ramp_vsrc and ramp_vsrc for more information on power supply ramping. When supply ramping is enabled and this parameter is not set, then the power supply ramping for leakage simulations will follow value set for the sim_init_duration parameter.

This parameter must be used before the `char_library` command.

library_copyright

<value> The string to denote copyright in the output library (Default: "")

This parameter specifies the value of the copyright attribute in the output library.

Note: The copyright can also be provided in the write_library `-user_data` file. The `-user_data` comment will take precedence over this parameter.

This parameter must be used before the `char_library` command.

Example

```
# Set the copyright line
set_var library_copyright "Cadence Design Systems, 2006-2013"
```

library_revision

<value> The string to use for denoting the library revision. Default: 1.0

This parameter sets the value of the revision attribute in the output library. Note: The revision can also be provided in the write_library `-user_data` file.

This parameter must be used before the `char_library` command.

Example

```
# Set the revision to 2.0
set_var library_revision "2.0"
```

library_revision_mode

`<0 | 1>` Identifies the format in which the `library_revision` text string should be inserted.
Default 1.

- `0` Inserts `library_revision` in the following format if the `library_revision` is specified as `"1.0"`:

  ```
  revision : "$Revision: 1.0 $";
  ```

- `1` Inserts `library_revision` in the following format if the `library_revision` is specified as `"1.0"`:

  ```
  revision : "1.0";
  ```

This parameter must be used before the `char_library` command.

lic_max_timeout

<value> Specifies the duration of time, in seconds, to wait for the required licenses to be acquired. Default: 86400 (seconds)

When starting up, Liberate will attempt to check out all licenses that are needed. For a Server, 1 server license is needed. For a client, Liberate will need 1 client license for each thread (see `char_library -thread`). If `ALTOS_QUEUE` is set and if only one license is needed, then Liberate will wait until a license is available and then start running. If `ALTOS_QUEUE` is set and more than 1 license is needed, then Liberate will wait until the timeout or it has all of the licenses it needs for all threads. Set the `lic_max_timeout` parameter to specify the number of seconds that Liberate will wait for licenses. When the timeout ends, if Liberate has at least 1 license, then it will stop waiting and start execution with however many licenses it has. If Liberate has no licenses at the end of the timeout, then it will reset the timeout clock and begin waiting again for licenses. After execution begins, Liberate will stop looking for additional licenses.

For example, if `ALTOS_QUEUE` is set to 1 along with `char_library -thread 4`, and if there are only 2 (mix-and-matched Liberate_Client, Variety_LX_Client and Liberate_LX_Client) licenses available at beginning, then Liberate will remain in a wait-and-check queue for an additional 2 licenses. As soon as the additional 2 client licenses are checked out successfully, Liberate will start execution with 4 simulation threads. If, however, there are no additional licenses checked out at the end of the timeout, then Liberate will start execution with only 2 simulation threads.

If ALTOS_QUEUE is set to 0 or is not set, then Liberate will not wait for licenses. Instead, it will check out as many licenses as it can (not exceeding the number it needs) and will begin execution. If no licenses are available, then Liberate will terminate.

The ALTOS_LIC_MAX_TIMEOUT shell environment variable will override the value set by this parameter in the Tcl file.

This parameter must be used before the char_library command.

lic_queue_timeout

<value>	Specifies the duration 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 will override the value set by this parameter in the Tcl file.

This parameter must be used before the char_library command.

logic_and

"*string*"	The characters to use for denoting logic AND in library attributes. Default: " * " *(Notice the space on both sides of *))*

This parameter sets the logic AND string for attributes that contain logic functions such as *when* conditions. It does not apply to sdf_cond attributes where the string for logic AND can be set by the sdf_logic_and parameter.

This parameter must be used before the char_library command.

logic_not

"*string*"	The characters to use for denoting logic NOT in library attributes. Default: "!"

This parameter sets the logic NOT string for attributes that contain logical functions such as *when* conditions. It does not apply to sdf_cond attributes where the string for logic NOT can be set by the sdf_logic_not parameter.

This parameter must be used before the char_library command.

logic_or

"*string*"	The characters to use for denoting logic OR in library attributes Default: " + " *(Notice the space on both sides of +)*

This parameter sets the logic OR string for attributes that contain logical functions such as *when* conditions. It does not apply to `sdf_cond` attributes where the string for logic OR can be set by the `sdf_logic_or` parameter.

This parameter must be used before the `char_library` command.

Example

```
# Set the logic AND, OR and NOT string to &&, ~, |
set_var logic_and "&&"
set_var logic_not "~"
set_var logic_or "|"
```

lpe_derate_mode

`< 0 | 1 >`	Specifies whether the derate flow should be enabled. Default: `0`	
	`0`	Disable the derate flow.
	`1`	Enable the derate flow.

The set_context command should be used along with `lpe_derate_mode=1`. If `lpe_derate_mode=0`, the `set_context` command does not work. In addition, this parameter must be used before the char_library command.

lvf_data_char_checks

`< 0 | 1 | 2 >`

Runs the Liberate LV `check_lvf_data -check char_checks` utility when the write_library `-lvf` command is set in Liberate or the `write_variation` command is set in Liberate Variety.

- `0` — Disable `check_lvf_data`.
- `1` — Enable `check_lvf_data` with non-verbose report.
- `2` — Enable `check_lvf_data` with verbose report.

Regular `check_lvf_data` report and Tcl script containing select_arc commands based on the custom tolerance outliers are written to the `<lib>.check_lvf_data.rpt` and `<lib>.check_lvf_data.select_arc.tcl` files, respectively.

The following custom tolerances are available:

```
set abstol {out_of_bound 8}
set reltol {ocv_within_cell 80 \
    ocv_within_cell_table_fail_ratio 0.5 \
    early_late_sigma_ratio 500 \
    delay_trans_sigma_ratio 20}
```

This parameter must be used before the write_library command is run for model creation.

lvf_enable_retain

`< 0 | 1 >` Requests Liberate to model LVF format data for retain arcs. Retain arcs are common in SRAM designs but not in standard cells. See the Liberty specifications for more details on retain arcs.
Default: `0`

The retain arc groups name are as following:

- `ocv_sigma_retaining_rise`
- `ocv_sigma_retaining_fall`
- `ocv_sigma_retain_rise_slew`
- `ocv_sigma_retain_fall_slew`

`0` Do not generate LVF format data for retain arcs.

`1` Generate LVF format data for retain arcs.

This parameter must be used before the write_library command is run for model creation.

mac_address_query_timeout

`<time_in_seconds>` Specifies a maximum time window to keep re-trying a given MAC address. Default: 60 (seconds)

The system will keep trying to access a MAC address, and if there is no response within the set time limit, the system will give up on that address. Useful in networks with heavy traffic.

-1: No timeout – keep trying the address until there is a response.
0: No retry – if there is no response on the first try, give up. (Behavior of versions 3.1 and older.)
Positive number: Maximum amount of time to continue re-trying. (Default = 60 seconds.)

This parameter must be used before the `char_library` command.

Example

```
# Set timeout to a minute and a half:
set_var mac_address_query_timeout 90
```

mark_failed_data

`< 0 | 1 >` Select the warning level for failed data. Default: 1

- `0` Does not issue any warnings. The output library contains the value as stored in the LDB. This value may be a legal number and may pass the library compilation without generating any warnings. In most cases, the constraint value in `.lib` is an exceptionally 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).

When Liberate determines that a characterization has failed, it sets the `altos_error_flag` inside the LDB. This parameter determines how Liberate will deal with the presence of this flag in the LDB.

This parameter must be used before the `write_library` command is run.

mark_failed_data_replacement

<string> Specifies the characters to use as replacement value in the output library for all the failed data. The `mark_failed_data` parameter must be enabled.
Default: "`1e+31`"

If mark_failed_data is set to `0`, then no non-constraint failed data is replaced. See constraint_failed_value for the handling of failed constraint data in this case.

This parameter must be used before the write_library command.

max_capacitance_attr_limit

<value>	Maximum allowed max_capacitance attribute value. Default: `1` (Farad)

Use this parameter to set a maximum value allowed for all `max_capacitance` attributes. This limit is applied after `max_capacitance_factor`.

This parameter can be used after the char_library command.

Example

```
# Set max capacitance attribute limit
set_var max_capacitance_attr_limit 1000e-15
```

max_capacitance_attr_mode

`< 0 | 1 >`	Controls how Liberate selects the `max_capacitance` attribute value. Default: `1`	
	`0`	Liberate will select the max of the maximum index_2 values.
	`1`	Liberate will select the min of the maximum index_2 (load_index) values. (Default and recommended)

This parameter must be used before the char_library command. However, this parameter will have no effect if the `write_library -derive_max_capacitance` option is used.

Example

```
set_var max_capacitance_attr_mode 1
```

max_capacitance_auto_mode

`< 0 | 1 | 2>` Selects the computation mode that determines the `max_capacitance` attribute for tie-high and tie-low cells. Default: `1`

- `0` Selects the original code that is part of the `auto_index` algorithm for determining the `max_capacitance` attribute for tie-high and tie-low cells.

  Note: This setting is provided for backward compatibility.

- `1` Divides the `max_transition` by the resistance drain to source (Rds) to calculate the `max_capacitance` attribute for tie-high and tie-low cells. This calculation is done when the `-auto_index` or `-auto_max_capacitance` option of the `char_library` and `write_vdb` commands is enabled.

- `2` Calculates the `max_capacitance` attribute for TIE cells without requiring the `-auto_index` or `-auto_max_capacitance` option of the `char_library` command.

To measure the Rds, connect a 100kOhm resistor to the opposite supply and compute the Rds as following:

```
TieHi_Ion = (Vdd -Vth)/Rds = Vth/Rl
TieLo_Ion = (Vdd -Vtl)/Rl = Vtl/Rds
Max_cap = max_transition / Rds
```

Where:

- `Vth` = The tie-high steady-state output pin voltage.
- `Vtl` = The tie-low steady-state output pin voltage.

This parameter must be used before the `char_library` command.

max_capacitance_derive_limit_maxload

`< 0 | 1 >`	Limits the max_capacitance attribute to the largest index_2 characterized load value. Default: 1

1: Limits the max_capacitance attribute to the largest index_2 characterized load value when write_library -derive_max_capacitance is enabled. (Default and Recommended)
0: Allows max_capacitance to be larger than the largest index_2 characterized value.

This parameter must be used before the `write_library` command.

max_capacitance_factor

<value>	Multiplication factor applied to all *max_capacitance* attributes Default: 1

This parameter can be used to apply a factor to all *max_capacitance* attributes in the library. This factor does not get applied to indexes. Default: 1.

This parameter must be used after the `char_library` command.

Example

```
# Set max capacitance factor
set_var max_capacitance_factor 0.66
```

max_capacitance_limit

<value>	Maximum allowable load index value. Default: *1 (Farads)*

This parameter controls a global limit that only becomes effective when `char_library -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 will be reset to the value stored in this parameter.

This parameter must be used before the `char_library` command.

Example

```
# Set maximum allowed capacitance load
set_var max_capacitance_limit 1e-9
```

max_hidden_vector

<value> Maximum number of hidden power vectors (states). Default: 64

This parameter sets the maximum number of vectors that will be characterized for hidden power. The hidden power vector count is determined from the internal nodes and primary pins such that the hidden power vectors are unique. The input pins, bi-directional pins, output pins, and internal nodes that can have unique states are included in determining the vector count.

Internal nodes such as intermediate nodes in a transistor stack and storage nodes can have unique states depending on their initial values. Default: 64.

This parameter must be used before the `char_library` command.

Example

```
# Set maximum number of hidden power vectors
set_var max_hidden_vector 1000
```

max_leakage_vector

<value> Maximum number of leakage vectors (states). Default: 256

This parameter sets the maximum number of vectors that will be characterized for leakage. The leakage vector count is determined from the internal nodes and primary pin such that the leakage vectors are unique. The input pins, bi-directional pins, output pins and internal nodes that can have unique states are included in the vector count determination. Internal nodes such as intermediate nodes in a transistor stack and storage nodes can have unique states depending on their initial values. Default: 256.

This parameter must be used before the `char_library` command.

Example

```
# Set maximum number of leakage vectors
set_var max_leakage_vector 1000
```

max_noise_width

<value>	Maximum allowable noise-glitch width (in seconds). Default: -1 (Don't automatically create indexes)

This parameter is used to enable automatic creation of the indexes for `si_immunity` tables. This parameter sets the maximum allowable noise-glitch width. Setting this parameter overrides indexes defined in noise immunity templates. This parameter should always be set when characterizing noise immunity.

This parameter must be used before the `char_library` command.

Example

```
# Set maximum noise glitch width to 4ns
set_var max_noise_width 4e-9
```

max_transition

<value>	Maximum allowable delay transition time (in seconds). Default: `3.0e-9` (3ns)

This parameter is used to limit the maximum allowable output transition for a cell. Set this parameter when using the `auto_index` option to `char_library`. The `define_max_transition` command can be used to specify a pin based local override for the max_transition.

This is the report value in the Liberty model. In the case where `slew_*` parameters differ from their `measure_slew_*` counterparts, Liberate will derive the slew used during simulation by multiplying `max_transition` by the value used for `slew_derate_from_library`.

This parameter must be used before the `char_library` command.

Example

```
# Maximum output transition time allowed
set_var max_transition 1e-9
```

max_transition_attr_limit

<value>	Maximum allowed `max_transition` attribute value. Default: `1` (Second)

Use this parameter to set a maximum value allowed for all `max_transition` (input, bidi and output pin) attributes written to the output library. If the maximum `index_1` (transition index) value exceeds the limit set by this parameter, the `max_transition` value will be overridden to this value.

This parameter is applied after the max_transition_factor parameter.

This parameter must be used before the write_library command.

Example

```
# Set max transition attribute limit to 1ns
set_var max_transition_attr_limit 1e-9
...
write_library my.lib
```

max_transition_factor

<value>	Multiplication factor applied to all `max_transition` attributes. Default: `1`

This parameter can be used to apply a factor to all `max_transition` attributes in the library. This factor does not get applied to indexes.

This parameter must be used after the `char_library` command.

Example

```
# Set max transition factor
set_var max_transition_factor 0.66
```

max_transition_for_outputs

`<-1 | value | max>` Requests that the `max_transition` attribute be modeled on cell output pins.
Default: `-1` (output pins do not have the attribute max_transition)

`-1`	Do not output the `max_transition` attribute into the output library on output pins.
`<value>`	Write the `max_transition` attribute into the output library with the specified value (in seconds).
`max`	Write the `max_transition` attribute into the output library. The value that is output is the maximum value of all the `rise_transition` / `fall_transition` data table values from all the arcs ending at that pin.

Note: The `max_transition` value will be limited by the max_transition_attr_limit parameter.

This parameter must be used before the `write_library` command.

Example

```
set_var max_transition_for_outputs max
...
write_library my.lib
```

max_transition_include_power

`<0 | 1>` Determines whether power arcs should be included in the `max_transition` counting.
Default: `0`

`0`	Do not consider the power arcs. The `max_transition` attribute is determined from timing arcs only.
`1`	Include the power arcs in the search for determining `max_transition`.

Note: Liberate will automatically determine the `max_transition` attribute from the characterized timing arcs. Some cells, such as keeper cells do not have any timing arcs.

This parameter must be used before the write_library command is run.

max_transition_include_rcvr_cap

`<0 | 1>` Determines whether the largest value of the receiver slew should be included in the `max_transition` counting.
Default: `0`

`0`	Do not include the receiver slew.
`1`	Include the receiver slew.

This parameter must be used before the write_library command is run.

measure_cap_active_driver_mode

`<0 | 1>` Instructs Liberate to begin measuring the pin capacitance from the time corresponding to the last supply crossover.

When an active driver is used to create the input waveform, the waveform shape can transition multiple times around the supply at the beginning of the input transition. When the pin capacitance measurement has a threshold close to the initial supply voltage, this parameter setting may result in a negative pin capacitance value.
Default: `0`

Note: This measurement change is applied to both ECSM and CCS pin capacitances.

- `0` Measures the pin capacitance from the beginning of the input waveform without observing the actual shape of the waveform.
- `1` Measures the pin capacitance from the last supply crossing at the beginning of the input transition.

This parameter must be used before the char_library command is run.

measure_cap_lower_fall

<value> The percentage point on the cell input waveform to use for measuring falling input capacitance to. Default: `0.01` (1% of supply)

measure_cap_lower_rise

<value> The percentage point on the cell input waveform to use for measuring rising input capacitance from. Default: `0.01` (1% of supply)

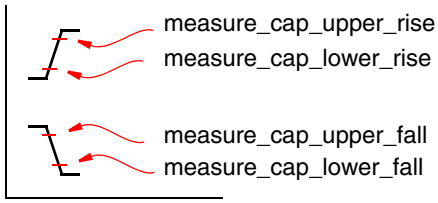
measure_cap_upper_fall

<value> The percentage point on the cell input waveform to use for measuring falling input capacitance from. Default: `0.99` (99% of supply)

measure_cap_upper_rise

<value> The percentage point on the cell input waveform to use for measuring rising input capacitance to. Default: `0.99` (99% of supply)

The above parameters are used to control how input capacitance is measured after SPICE simulation.



These parameters must be used before the char_library command is run.

Examples

```
# Set the capacitance measurements to 10-90%
set_var measure_cap_lower_fall 0.1
set_var measure_cap_upper_fall 0.9
set_var measure_cap_lower_rise 0.1
set_var measure_cap_upper_rise 0.9
```

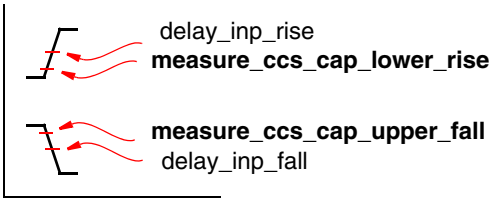
measure_ccs_cap_lower_rise

<value> The percentage point on the cell input waveform to use for measuring rising input CCS capacitance from. Default: `0.0` (0% of supply)

measure_ccs_cap_upper_fall

<value> — The percentage point on the cell input waveform to use for measuring falling input CCS capacitance from. Default: `1.0` (100% of supply)

These parameter are used to measure the first portion (C1) of the CCS receiver capacitance model; used in combination with the delay rise/fall thresholds:



This parameter must be used before the `char_library` command.

Example

```
# Set the CCS capacitance measurements
set_var measure_ccs_cap_upper_fall 0.3
set_var measure_ccs_cap_lower_rise 0.7
```

measure_em_target_occurrence

<last | integer>

Specifies any legal value that can be used in a SPICE measure statement to direct the measurement to use a specific target (TARG) match. This parameter only works with EM characterization (see `-em` of `char_library`).
Default: `1`

This parameter must be set before the `char_library` command.

Example

```
set_var measure_em_target_occurance last
```

measure_output_range

`< 0 | 1 >` Enables measurement of output transition range. Default: 0 (Assume full rail voltage swing)

This parameter impacts the measurement of the initial and final voltages of a pin. If this parameter is set then the measurement thresholds are applied to the range between the initial voltage and the 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 will be used. The default is `0` that means the outputs are assumed to swing full rail. This parameter applies only when the `-io` argument to `char_library` is used.

This parameter must be used before the `char_library` command.

measure_output_range_abstol

<minimum_range_Volts>

Define the minimum output voltage swing range.
Default: `0.100` (Volts)

This parameter is used to specify the minimum supported output voltage swing range. It is enabled when the `measure_output_range=1` parameter. If the measured output swing range is less than the value specified in this parameter, then the value specified in this parameter is used.

This parameter must be used before the `char_library` command.

measure_slew_lower_fall

<value> The percentage point on the cell output waveform to measure falling output transition times to. Default: `0.2` (20%)

measure_slew_lower_rise

<value> The percentage point on the cell output waveform to measure rising output transition times from. Default: `0.2` (20%)

measure_slew_upper_fall

<value> The percentage point on the cell output waveform to measure falling output transition times from. Default: `0.8` (80%)

measure_slew_upper_rise

<value> The percentage point on the cell output waveform to measure rising output transition times to. Default: `0.8` (80%)

The above parameters are used to control how output transition times are measured after SPICE simulation.

These parameters must be used before the `char_library` command.

Examples

```
# Set the transition measurements to 20-80%
set_var measure_slew_lower_fall 0.2
set_var measure_slew_upper_fall 0.8
set_var measure_slew_lower_rise 0.2
set_var measure_slew_upper_rise 0.8
```

measure_target_occurrence

`<last | legal_spice_value>`

Specify any legal spice measure value. Default: `last` (assume full rail voltage swing)

When dual outputs are tied together by a resistor, oscillations can occur. Set this parameter to any legal value that can be used in a SPICE measure statement to direct the actual measurement to use a specific match.

This parameter must be used before the `char_library` command.

mega_analysis_mode

`<0 | 1 | 2 | 3>` Controls how the *Inside View* algorithm analyzes a circuit and breaks it into components for vector generation.
Default: `0`

- `0` Specifies to use the normal `mega` analysis mode. This setting disconnects components separated by *off* transistors.
- `1` Does additional node function pruning for cases that are not handled well by mode `0`. This pruning might be over-aggressive in general. Therefore, it should be used only for multi-bit flip-flop cells that require it.

  Note: It is not recommended to combine the define_cell `-when` command with `mega_analysis_mode=1`. Restricting the logical states in this way can result in a pruned function that does not allow all legal arcs.
- `2` Provides experimental function pruning.

  Note: Use of this value is not recommended.
- `3` Specifies to use define_bundle_pins to restrict cross-bit function dependencies.

This parameter must be used before the `char_library` command.

Example

```
set_var -cell *ELP* mega_analysis_mode 1
```

mega_bundle_mode

`<0 | 1 | 2>` Controls how mega mode should use the information from `define_bundle_pins`. Default: `1`

- `0` Do not consider bundle pins while generating the vectors. This means that when the `mega_bundle_mode` parameter is set to `0`, Liberate will not use the information from the `define_bundle_pins` parameter.
- `1` Use bundle pins to restrict all vector settings for side pins.
- `2` Use bundle pins to restrict leakage vector settings with side pins.

This parameter must be used before the `char_library` command.

mega_enable

`<2 | 3>` Enables the use of an advanced algorithm referred to as "mega" mode. This mode improves the *Inside View* algorithm preprocessing of large cells.
Default: `2`

- `2` Enables mega mode for all cells in the library. It is not necessary to use the `define_cell -type mega` option.
- `3` Uses the mega mode algorithm on a cell by cell basis and is enabled only when the `define_cell -type mega` option is used for a specific cell.

Note: `0` and `1` are unsupported modes.

This parameter must be used before the `char_library` command.

mega_floating_node_reduction

`<0 | 1 | 2>`

Controls how to handle floating nodes such as found in level shifters or some power-down designs.
Default: `0`

- `0` — Ensures that floating nodes are expanded to have both values `0` and `1`.
- `1` — Ensures that floating nodes are treated as don't care. This prevents the 2^n number of vectors for `n` floating nodes, but may cause a loss of accuracy for cases where the initial value of the floating node strongly affects delay times.
- `2` — Same as `1`, but only for cells that use define_bundle_pins (or define_bus) to restrict the vectors.

This parameter must be used before the `char_library` command.

mega_mode_constraint

`< minimum | fanout | all >`

Enables heuristics that controls the number of combinations (vectors) generated for the side pins when characterizing a constraint arc.
Default: `fanout` (which is numerically represent by `1`)

- `minimum | 0` — Specifies to select a single vector for all the side pins. If possible, cell output that are not on the characterized path are set to be non-switching. This gives the best runtime by using one arbitrary combination of side pins.

`fanout | 1`	Specifies to select all combinations of side pins that can have some effect on the measured constraint. This includes the nodes that are in the path from input to probe for the arc being characterized and all nodes that are one Channel Connected Component (CCC) away from this path (first order miller effect). This mode covers all loading conditions for the constrained path and provides the best constraint accuracy. For multi-bit cells this mode may produce an exponential number of vectors if not restricted by the `-patterns` option of the `define_bundle_pins` command.
`all | 2`	Specifies to generate simulation vectors for all combinations of side pins. This mode is intended for maximum compatibility with non-mega (when `mega_enable` is equal to `3`) results. For multi-bit cells this mode produces the longest runtime due to the maximum number of vectors applied. The vectors can be restricted by the `-patterns` option of the `define_bundle_pins` command.

Mega mode (see mega_enable) identifies the cone of logic from inputs to probe for each constraint arc to be characterized. The signals that do not impact logically the constraint arc are considered side pins. These side pins may be completely independent of the cone of logic or they may have a parasitic effect on the arc. When possible the vector_side_input setting is observed.

This parameter must be used before the `char_library` command.

mega_mode_delay

`< minimum | fanout | all >`

Enables heuristics that controls the number of combinations (vectors) generated for the side pins when characterizing a delay arc. Default: `all` (which is numerically represent by `2`)

`minimum | 0`	Select a single vector for all the side pins. If possible, cell outputs not on the characterized path are set to be non-switching. This gives the best runtime by using one arbitrary combination of side pins.
`fanout | 1`	Select all combinations of side pins that can have some effect on the measured delay or power. This includes all nods that are switching (due to their effect on power). This mode covers all loading conditions for the delay path and provides the best accuracy. For multi-bit cells this mode may produce an exponential number of vectors if not restricted by the `-patterns` option of the `define_bundle_pins` command.
`all | 2`	Generate simulation vectors for all combinations of side pins. This mode is intended for maximum compatibility with non-mega (when `mega_enable` is equal to `3`) results. For multi-bit cells this mode produces the longest runtime due to the maximum number of vectors applied. The vectors can be restricted by the `-patterns` option of the `define_bundle_pins` command.

Mega mode (see mega_enable) identifies the cone of logic from inputs to output for each combinational (see define_arc -type combinational) arc to be characterized. The signals that do not impact the logic for this delay arc are considered side pins. These side pins may be completely independent of the cone of logic or they may have a parasitic effect on the arc. When possible the vector_side_input setting is observed.

This parameter must be used before the `char_library` command.

mega_mode_hidden

`< minimum | fanout | all >`

Enables heuristics that controls the number of combinations (vectors) generated for the side pins when characterizing a hidden power arc.
Default: `fanout` (which is numerically represent by `1`)

`minimum | 0`	Select a single vector for all the side pins. This gives the best runtime by using one arbitrary combination of side pins.
`fanout | 1`	Select all combinations of side pins that can have some effect on the hidden power. This includes all nodes that are switching (due to their effect on power). This mode covers all loading conditions for the hidden power and provides the best accuracy. For multi-bit cells this mode may produce an exponential number of vectors if not restricted by the `-patterns` option of the `define_bundle_pins` command.
`all | 2`	Generate simulation vectors for all combinations of side pins. This mode is intended for maximum compatibility with non-mega (when `mega_enable` is equal to `3`) results. For multi-bit cells this mode produces the longest runtime due to the maximum number of vectors applied. The vectors can be restricted by the `-patterns` option of the `define_bundle_pins` command.

Mega mode (see mega_enable) identifes the cone of logic from inputs internal node for each hidden power arc (see `define_arc -type hidden`) arc to be characterized. The signals that do not impact the logic cone for this hiden power arc are considered side pins. These side pins may be completely independent of the cone of logic or they may have a parasitic effect on the arc. When possible the vector_side_input setting is observed.

This parameter must be used before the char_library command is run.

mega_reduced_function_mode

`< 0 | 1 >` Determines how to reduce the per-node function to include only logical dependencies for use in determining hidden vectors.
Default: `1`

- `0` Use the standard function and do not reduce the per-node function.
- `1` Use a separate circuit trace to determine the reduced function.

This parameter must be used before the char_library command is run.

mega_reverse_mos_mode

`< 0 | 1 >` Controls whether the *Inside View* algorithm (mega mode) should be allowed to find paths to ground nets through PMOS and to power nets through NMOS.
Default: `0`

- `0` Allows mega mode to find paths to ground nets through PMOS and to power nets through NMOS.
- `1` Prevents mega mode from finding paths to ground nets through PMOS or to power nets through NMOS.

  For some cells, such as those in sleep transistor designs, a component (like a turn-off transistor) is shared among all channel-connected blocks (CCBs). This results in sneak paths from one CCB to another causing real but unintended behaviors compared to an equivalent cell without the shared component. For these cases, setting the `mega_reverse_mos_mode` parameter to `1` simplifies the function analysis by keeping the CCBs separate.

This parameter must be used before the char_library command is run.

mega_short_circuit_mode

< 0 | 1 | 2 >

Controls how the *Inside View* algorithm handles the short-circuit leakage states. These are the states in which an internal transistor gate is simultaneously pulled high and low for some input pin combination. An example of this can be a buffer with complementary inputs AN and AP where input combination AN*!AP would produce a short-circuit.
Default: 1

- 0 — Ignore short-circuit state as invalid. This is the same behavior as was supported before LIBERATE 19.2.
- 1 — Warn about short-circuit states that are ignored. For example,

```
WARNING (LIB-819): Omitting short-circuit
vectors for cell 'BUF', state 'AN=1*AP=0'.
Check that the circuit and netlist are correct.
To allow the short-circuit state to be
characterized, set parameter
'mega_short_circuit_mode' to '2' and rerun the
Tcl script.
```

- 2 — Allow short-circuit states for leakage. As a result, the short-circuit nodes are not initialized in the deck.

This parameter must be used before the char_library command is run.

merge_related_preset_clear

< 0 | 1 | 2 >

Enables the merging of related preset and clear states.
Default: 2

The assertion of an asynchronous pin (see define_cell -async) will cause a clear or preset half-unate arc to be characterized. The de-assertion of a preset/clear signal (see the parameter combinational_risefall) may also characterize a half-unate arc. An arc related to de-assertion can have a timing_type attribute of combinational or clear/preset type. An example of this is a sequential cell with both a set and clear pin where both are asserted and one is de-asserted. These two half-unate timing arcs have a different timing type and can be modeled separately or merged into a single timing group.

For the merging to work properly, the cell function must correctly represent the circuit operation. If Liberate fails to determine the correct function for the cell, you must provide the logical function in the user_data file. That is, if Liberate does not produce a valid function for the cell, then you must provide a valid function.

- 0 — Do not merge two asynchronous arc half-unate timing groups for a given related_pin together into a single timing group. The library will contain two half-unate timing groups, one with a combinational timing type and one with a clear/preset timing type.
- 1 — Merge the rise and fall tables with different timing_type attributes from a clear/preset asynchronous related_pin into a single timing group with the timing_type set to clear/preset.
- 2 — Do not merge two asynchronous arc half-unate timing groups as with the setting of 0 above. However, both half-unate timing groups will be assigned a timing_type attribute of clear or preset instead of one having a timing_type attribute of combinational.

This parameter can be used before the write_library command.

Example

```
# Allow the characterization and merging of asynchronous arc half-unate
# timing types into a single timing group with a clear/preset timing_type.
set_var combinational_risefall 1
set_var merge_related_preset_clear 1
```

min_capacitance_for_outputs

`<0 | 1>`	Enables min_capacitance attribute for output pins. Default: `0`

Set this parameter to a `1` to enable the output of the *min_capacitance* attribute on output pins. A setting of `1` is useful to match legacy libraries. When set to `0` (default), Liberate will not output the *min_capacitance* attribute on output pins.

This parameter must be used before the `char_library` command.

min_output_cap

`<value>`	Minimum allowable output capacitance (in Farads). Default: `-1` (use min input cap found in library)

This parameter is used to set the minimum output capacitive load when using the `auto_index` option with `char_library`. If not specified the minimum input pin capacitance found in the library is used.

This parameter must be used before the `char_library` command.

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_library -auto_index
```

min_period

`<0 | 1 | 2>`

Controls the calculation of the `minimum_period` timing group for clock and asynchronous pins. The `minimum_period` is calculated as 2 * max (mpw_high, mpw_low)
Default: `0`

Note: Modeling of `min_period` is dependent on the MPW data that is characterized and stored in the ldb. If MPW is not characterized, then `min_period` cannot be computed.

`0`	No minimum period calculation.
`1`	Calculates minimum period for clock (flip-flops) and enable (latch) pins only
`2`	Calculate minimum period for clock/enable pins plus async pins.

This parameter must be used before the `write_library` command.

min_period_when

`"string"` Specifies the logic *when* condition for the `min_period` constructs. Default: ""

Setting this parameter adds both a *when* condition and an equivalent *sdf_cond* condition to the `minimum_period` group. Only *min_pulse_width* groups that overlap with the user-defined *when* condition will be used to calculate the `minimum_period`.

This parameter works only in conjunction with the `min_period` parameter.

This parameter must be used before the `write_library` command.

min_transition

<value>	Minimum allowable delay transition time (in seconds). Default: -1 (automatically calculate)

This parameter is used to set the minimum output transition when using the auto_index option to char_library. Default is to automatically calculate the minimum transition index.

This parameter must be used before the char_library command.

min_transition_attr_limit

<value>	Minimum allowed min_transition attribute value. Default: 1e-15 (seconds)

Use this parameter to set a minimum value allowed for all min_transition attributes. This limit is applied after min_transition_factor. (See write_min_transition_attr to enable writing the min_transition attribute into the library.)

This parameter can be used after the char_library command.

Example

```
# Set min transition attribute limit
set_var min_transition_attr_limit 1e-12
```

min_transition_factor

<value>	Multiplication factor applied to all min_capacitance attributes. Default: 1

This parameter can be used to apply a factor to all min_transition attributes in the library. This factor does not get applied to indexes. Default: 1.

This parameter can be used after the char_library command.

Example

```
# Set min transition factor
set_var min_transition_factor 0.66
```

min_transition_for_outputs

`<-1 | "min" | value>`

Adds the `min_transition` attribute on output pins. Default: `-1` (Do not add this attribute.)

`-1`	Do not put the attribute `min_transition` on outputs.
`min`	Use the minimum value of all the `rise`/`fall_transition` values from all the arcs ending at that pin.
`value`	Forces a specific `min_transition` value on an output pin.

This parameter must be used before the `write_library` command.

min_transition_include_power

`<0 | 1>`

Controls whether to include power arcs when determining `min_transition`. Default: `0`

`0`	Ensures that `min_transition` is determined from timing arcs only. Liberate will automatically determine the `min_transition` attribute from the characterized timing arcs. Some cells, such as keeper cells do not have any timing arcs.
`1`	Instructs Liberate to include power arcs in the search for determining `min_transition`.

This parameter must be used before the `write_library` command.

minimum_memory_warning_limit

<value>	Specifies the minimum disk space in GB. If the disk space is between the specified value and 10 MB, Liberate issues a warning message. If the disk space is less than 10 MB, the job being processed at the client is stopped.

model_vth

<0 | 1 | 2 |3>	Controls how to model threshold voltage (Vth) in the library. Default: 1

0	Do not write Vth to the library.
1	Write Threshold Voltage and Maximum Transconductance (vtgm) to the library. Vtgm is calculated using the maximum gm method. Before modeling, enable running of the Vth simulation using the set_vtgm_cell command.
2	Write Vth to the library. Vth is the modeled Vth that the simulator prints according to the device parameters. Before modeling, enable running of the Vth simulation using the set_vtgm_cell command.
3	Write both vtgm and Vth to the library.

Example

```
#Enable Vth simulation using INVX1 as a test cell. Then write Vth to the library.
set_vtgm_cell -cells {INVX1}
set_var model_vth 2
```

mpw_criteria

<delay | glitch>	Specifies MPW failure criteria as delay or glitch. Default: delay

This parameter can be used to specify the failure criteria to be used when measuring the minimum pulse width. To measure Minimum Pulse Width (MPW) Liberate applies a narrowing pulse to the input pin until the failure criteria is met. The circuit is sensitized to enable a CLK-

>Output delay measurement. Legal values are: `delay` and `glitch`. When `delay` is specified, the `constraint_delay_degrade` parameter specifies the relative delay degrade value for the CLK->Output arc. When `glitch` is specified, the CLK->Output delay is measured at the `constraint_glitch_peak` ratio of supply and the searches for a pass/fail of the CLK->Output arc, that is, the CLK->Output path returns a measurement (it toggles) or the measurement fails (the Output does not toggle) when measured at the `mpw_glitch_peak` threshold.

This parameter must be used before the `char_library` command.

mpw_glitch_peak

`<ratio>`	Specify the MPW glitch peak failure threshold. Default: `0.95`

Set this parameter to the desired measurement threshold when the Minimum Pulse Width (MPW) measurement uses the glitch criteria (see mpw_criteria). This parameter has no effect if the `mpw_criteria` is set to `delay`.

This parameter must be used before the `char_library` command.

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. Default: `0.995 (99.5% of Vdd)` **Note:** This default value enables correlation between LVF and Monte Carlo flow.

This parameter must be used before the `char_library` command.

mpw_linear_waveform

`<0 | 1>` Use linear input waveform for MPW measurements.
Default: `1`

- `0` Use the same input waveform as for delay.
- `1` Use the linear input waveform.

Set this parameter to request that Liberate use a linear waveform on the input pin when characterizing the minimum pulse width. When disabled, the same input waveform used for delay is also used for MPW measurements. By default, a linear input waveform is used for MPW measurements.

If mpw_input_threshold is less than `1` (that is, it allows a pulse that does not reach the rail), the pulse is constructed from the initial portion of the first edge and the final portion of the second edge with the peak at the point where the two waveforms cross. This is true for both linear and piece-wise linear (PWL) waveforms.

When using `mpw_linear_waveform=0`, it is sometimes useful to reduce the `mpw_input_threshold` value if the PWL waveform has a long tail such that the default value (`0.9`) would result it an unnecessarily large lower bound on the MPW.

This parameter must be used before the `char_library` command.

mpw_related_output_pin

<string> Specifies the output pin or pins to use for the `related_output_pin` attribute if two-dimensional MPW is being used. If no pin name is specified, an arbitrary output pin is chosen. If multiple outputs are specified, a separate two-dimensional MPW table is created for each. This parameter can be set per-cell. A simple wildcard ("*") is supported.
Default: `" "`

This parameter must be used before the `char_library` command.

Examples

```
set_var mpw_related_output_pin q
set_var mpw_related_output_pin *
set_var mpw_related_output_pin {q qn}
```

```
set_var mpw_related_output_pin {q* so}
```

mpw_search_bound

<value>	Controls the initial search bound for MPW characterization. Default: 5e-9

This parameter must be used before the char_library command.

mpw_search_mode

<0 | 1 | 2>	Enables fast algorithm for determining MPW. Default: 1	
	0	Default algorithm using bisection techniques.
	1	Enables a fast algorithm for determining minimum pulse width. This may produce a slight change in results, but they should be within tolerance. (See constraint_search_time_abstol.) (Default)

2	This mode enables a search method which requires that the Minimum Pulse Width (MPW) exceeds the `related_pin` to probe delay. Use this method to increase risk aversion while significantly decreasing the run time. For this method to be successful, the mpw_criteria must use delay and the probe must be selected so the leading edge of the pulse is the active edge. This search method measures the `related_pin` to probe delay and adjusts the pulse width so that the pulse trailing edge transition aligns with the probe transition. The search terminates when the difference between the two edges is less than the constraint tolerance. The probe is selected from the nodes that transition due to the pulse leading edge but not the trailing edge, as this is necessary for the search criterion to work. The probe nodes that are triggered from the pulse trailing edge, or with both edges, are not monitored. For table based MPW (see mpw_table), the fastest slew is characterized for MPW. The rest of the slews are calculated as the fastest slew MPW plus the difference between the slews. The constraint_check_final_state and constraint_check_rebound parameters can be used with this method.

This parameter must be used before the `char_library` command.

mpw_skew_factor

`<value>`	Enables skewing the ratio of the input rise versus input fall slew for calculating *min_pulse_width* constraints. Default: `1.0` (rise and fall input slews are identical)

This parameter must be used before the `char_library` command.

mpw_slew

`<value | min | mid | max>`

Defines the input slew for minimum pulse width characterization.
Default: `min`

This parameter specifies the value for the input slew to be used when determining the minimum pulse width timing constraint of a clock or async signal. The value can be any floating point number in seconds. If `min`, `mid`, or `max` is used then the minimum, middle or maximum input slew value is taken from the input slew indexes 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).

This parameter must be used before the `char_library` command.

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>`

Defines a ratio of the mpw_slew to apply to all clock nets.
Default: `1` (use mpw_slew)

This parameter specifies the ratio to apply to the `mpw_slew` when characterizing the minimum pulse width value of clock nets. By default, clock nets will use the slew determined when applying mpw_slew. When this parameter is set, the slew applied to clock nets will be: `mpw_slew_clock_factor * mpw_slew`.

This parameter must be used before the `char_library` command.

Example

```
# 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>` Controls generation of a one-dimensional table of the minimum pulse-width (MPW) values.
Default: `1`

- `0` Disables generation of a table of the MPW values.
- `1` Enables generation of a table of the MPW values. When this parameter is set, the MPW data table uses the `index_1` table as specified by the `constraint_template` associated with the `define_cell` command for the cell.

  Note: Characterization of the MPW tables increases the run time.

Use this parameter to enable the generation of a minimum pulse-width data table.

This parameter must be used before the `char_library` command.

mpw_vector_bin_mode

`<0 | 1 | 2 | 3>` Controls pre-check index point for binning vectors.
Default: `1` (First index point)

- `0` No binning.
- `1` First index point.
- `2` Middle index point.
- `3` Last index point.

Set this parameter to reduce the number of vectors used to characterize MPW on an input pin (which should reduce the characterization run time). Liberate performs the simulation needed to fill in a selected index in the table, and uses that to determine how vectors can be "binned" (consolidated). This parameter specifies which index point in the table should be

used to determine the binning of vectors.

This parameter must be used before the `char_library` command.

Example

```
set_var mpw_vector_bin_mode 2
```

msg_level

`<0 | 1>`	Controls the verbosity of error and warning messages. Default: `0`
`0`	Output error messages and useful warning and informational messages.
`1`	Output all messages. Caution: this setting can output a lot of messages (some of which may not be helpful) making it difficult to determine which messages are important.

This parameter must be used before the `char_library` command.

msg_level_user_data_override

`<0 | 1>`	Controls the verbosity messages related to inserting user data in the library. Default: `0`
`0`	No messages are displayed.
`1`	Display a message when an attribute named in the `user_data_override` parameter is overridden by data in the `write_library -user_data` file.

This parameter must be used before the `write_library` command.

msg_limit_per_type_per_cell

`<integer>` Controls the output limit of the maximum number of messages for each message type.
Default: `5`

This parameter must be used before the char_library command is run.

multi_pvt_incremental_flow

`<0 | 1>` Controls the enabling of incremental characterization flow.
Default: `0`

- `0` Runs the regular characterization flow.
- `1` Enables the incremental characterization flow.

This parameter must be used before the read_ldb command.

This incremental flow run using the `multi_pvt_incremental_flow` parameter is completely different from the `read_ldb -incremental` flow. A combined usage of both the flows is not supported.

The incremental flow tries to identify automatically any arcs that need to rerun based on the following criteria:

- If the netlist file changed, then the entire cell is rerun.
- If the model file changed, then all the cells are rerun.
- If individual arcs are modified, added, or removed, then they are run individually.

The flow runs a counting job for all the cells to determine the changes and based on the findings, either the arcs are rerun or the assembly job is run to generate the LDB and the cell-level library. This happens for all cells.

The flow involves the following stages in the given sequence:

1. Run a fresh characterization.
2. Run the recovery flow.
3. Get a full LDB.
4. Run the incremental flow.

Once a fresh characterization run is finished, to do the incremental run, use the following commands:

```
set_var multi_pvt_incremental_flow 1  #Has to be set before the read_ldb command
read_ldb /path/to/fresh_run
.....
.....
```

Currently, the following items are supported in the incremental characterization flow:

- M1PVT (though MNPVT runs, but it is not supported currently and detection of out-of-date arcs will be incorrect)
- Add, modify, and delete arcs using `define_arcs`
- Add, modify, and delete index values
- Modify the model files (only top-level model file is checked, not the include ones)
- Modify the netlist files (only top-level netlist file is checked, not the include ones)
- Add cells to the existing set of cells

Limitations:

- MNPVT flow is not supported.
- Recovery flow and incremental flow are not supported together. Only one flow can be run at a time.
- Post-characterization distribution is not supported when incremental flow is enabled.

multi_pvt_rechar_arc_ids

`<arc_ID_list>`	Specifies a list of unique arc IDs that need to be rerun for a cell and PVT combination. Default: `""` This parameter is used in the Liberate Trio Multi-PVT (define_pvt-based) flow when the multi_pvt_recovery_flow parameter is set to `1` to enable the read_ldb-based recovery flow.

This command must be used before the `read_ldb` command is run.

Use the following syntax to specify the list of arc IDs for recharacterization:

```
set_var -cell {<cell_names_list>} -pvt {<pvt_names_list>} \
    multi_pvt_rechar_arc_ids <arc_ID_list>
```

Unique arc IDs assigned to each arc are present in the LDB files as `altos_arc_index` attributes as shown in the following example:

```
cell (BUF) {
    leakage_power () {
        altos_arc_id : "arc1";
        altos_arc_index : "4";
    }
leakage_power () {
    altos_arc_id : "arc2";
    altos_arc_index : "5";
   }
pin ("Y") {
    timing ()  {
        related_pin : "A";
        cell_rise (delay_template) {
             altos_arc_id : "arc4";
             altos_arc_index : "0";
    }
}
output_ccb (BUF_Y_1) {
    dc_current (ccsn_dc_template) {
        altos_arc_id : "ccsn6";
        altos_arc_index : "2";
    }
    output_voltage_fall () {
        altos_arc_id : "ccsn7";
        altos_arc_index : "2";
    }
   }
 }
}
```

Example

```
set_var multi_pvt_recovery_flow 1
set_var -cell DFF -pvt * multi_pvt_rechar_arc_ids [list 1 2 3 5] ; # will rerun only arc IDs 1, 2, 3, 5 for cell DFF for all pvts. Rest of the arcs will be copied from the previous run as is.
read_ldb LDB.ldb.gz
```

multi_pvt_rechar_do_preprocessing

`<0 | 1>` Controls preprocessing of the cells during recharacterization in the recovery flow.
Default: `0`

This parameter is used in the Liberate Trio Multi-PVT (define_pvt-based) flow when the multi_pvt_recovery_flow parameter is set to `1` to enable the read_ldb-based recovery flow.

`0`	Disable preprocessing of all cells.
`1`	Runs preprocessing for the specified list of cells.

This command must be used before the `read_ldb` command is run.

By default, the preprocessing of the cells including the vector database generation is done only once in the first session. It is not repeated in the subsequent sessions to save the preprocessing time that could potentially be high for MBFF, high fanin, or complex cells. However, if you are making changes in the template or arcs, or are expecting that your changes will impact the vector database, it is recommended that you set the `multi_pvt_rechar_do_preprocessing` parameter for those specific cells.

This parameter cannot be set globally. Use the following syntax to specify a restrictive list of only the required cells:

```
set_var -cell {<cell_names_list>} multi_pvt_rechar_do_preprocessing  <0  |  1>
```

Example

```
set_var multi_pvt_recovery_flow 1
set_var –cell DFF* multi_pvt_rechar_do_preprocessing 1 ; # Run preprocessing for DFF cell
read_ldb LDB.ldb.gz
```

multi_pvt_recovery_flow

<0 | 1>	Controls whether the recovery flow should be enabled in the Liberate Trio Multi-PVT (define_pvt-based) flow. Default: 0
0	Disables the multi-PVT recovery flow.
1	Enables the multi-PVT recovery flow. This setting is required in the read_ldb-based recovery flow.

This parameter is for enabling the read_ldb-based recovery characterization flow in Liberate Trio Multi-PVT (define_pvt-based) flow. When this parameter is set and the previous LDB is read, Liberate automatically recovers all those arcs, cells, and PVTs that were reported as failed in the previous session. In addition, the tool attempts to rerun relevant characterization in the current session.

This command must be used before the read_ldb command is run. When running the recovery flow, set this parameter to 1 before reading the LDB.

The recovery flow involves the following steps:

1. Find out all the arcs that had an error when run the last time.
2. Find out all the arcs that did not run last time because the session was killed abruptly.
3. Recharacterize all the cell and PVTs that are specified using the multi_pvt_recovery_rechar parameter.
4. Recharacterize all the individual arcs for cell and PVT combination that has been specified using the multi_pvt_rechar_arc_ids parameter.
5. Rerun the preprocessing job for all the cells that have been set using the multi_pvt_rechar_do_preprocessing parameter.

Example

```
set_var multi_pvt_recovery_flow 1
read_ldb LDB.ldb.gz
```

multi_pvt_recovery_rechar

<0 | 1>	Controls recharacterization of the specified cells and PVTs irrespective of whether they passed or failed in the previous session. Default: 0 This parameter is used in the Liberate Trio Multi-PVT (define_pvt-based) flow when the multi_pvt_recovery_flow parameter is set to 1 to enable the read_ldb-based recovery flow.

Use the following syntax to specify the list of cells and PVTs for recharacterization:

```
set_var -cell {<cell_names_list>} -pvt {<pvt_names_list>} \
    multi_pvt_recovery_rechar  <0 | 1>
```

During this flow, you can also change the characterization settings as required. The modified settings are then used during recharacterization.

This command must be used before the `read_ldb` command is run.

Example

```
set_var multi_pvt_recovery_flow 1
set_var –cell DFF* -pvt FF* multi_pvt_recovery_rechar 1
read_ldb <LDB>.ldb.gz
set_var leakage_merge_state 2 ;
```

Recharacterizes all cells matching `DFF*` at all PVTs matching `FF*` and failed leakage arcs for any cells at other PVTs in the previous run.

net_batch_mode

`<0 | 1>` Specifies Netbatch as a queuing system. This parameter cannot be used together with `qsub_no_shell_mode`.
Default: `0` (Do not use Netbatch.)

nldm_measure_output_range

`<0 | 1>` Reports the `output_signal_level_*` attributes under the timing arc if they differ from the rails.
Default: `0` (disabled)

Some high-leakage cells can have arc-based partial output swings, where the waveforms might not start or end at the rail. Setting this parameter allows Liberate to capture and report the offsets in `output_signal_level_low` and `output_signal_level_high` under the timing arc instead of as a pin-based default.

- `0` Disables the reporting.
- `1` Enables the reporting.

See ecsm_measure_output_range for similar functionality for ECSM models.

This parameter must be used before the char_library command is run.

nochange_mode

<0 | 1 | 2>

Enables modeling and characterization of nochange timing type arcs.
Default: 0
Recommended: 1

0 — The nochange arcs are not characterized but can be modeled with user-specified delay values. You can specify the nochange timing arcs with the define_arc command. The value is specified by the -value argument of the define_arc command or by the nochange_value parameter. This setting is compatible with previous releases of Liberate.

Note: If a nochange arc cannot be characterized, the mode 0 can be used to specify the desired value.

1 — The nochange arcs are characterized by Liberate if a define_arc command is provided for the arc.

2 — Similar to 1. However, both pull-in and push-out are measured for both the leading and trailing output pulse edges instead of push-out for the leading edge and pull-in for the trailing edge.

This parameter must be used before the char_library command is run.

The nochange_mode parameter supports the -stage variation option. The value set with this setting only effects the variation simulation. For nominal data, setting the nochange_mode parameter to 2 is good because the final constraint is dependent upon the leading or trailing edge. However, during variation characterization, setting this parameter to 2 can cause Liberate Variety to incorrectly pick the difference between the leading or trailing edge as part of the sensitivity. Therefore, for variation characterization, set nochange_mode to 1 and use the -stage option.

Examples

```
set_var nochange_mode 2
set_var -stage variation nochange_mode 1
```

nochange_value

<value>	Specify a default nochange arc delay in seconds. Default: 300e-12 seconds

When nochange_mode is set to 0, this parameter provides the default delay value to be specified for the nochange arcs. This can be overridden by the -value argument of the define_arc command.

normalized_driver_waveform

{ list }	Use a given PWL waveform as the input driver waveform. Default: "" (Unused).

This parameter enables using a user-defined PWL waveform as the reference waveform for all input pins. The reference waveform is linearly scaled to match the slew and voltage range required for a given input slew.

The PWL must be entered in a normalized, ordered list of time-voltage pairs { t0 v0 t1 v1 ... tN vN } where t0=v0=0 and vN=1.0. Liberate automatically derives the reference slew from the reference waveform using the derating from given measure_slew_*_rise and slew_*_rise settings.

The default is an empty list. If set, it overrides any set_driver_cell commands and the predriver_waveform parameters.

While characterizing NLDM delay models, you can use this parameter for defining the input waveform shape. For detailed information, see Delay Models in Chapter 7, “Performing Characterization using Liberate.”

This parameter must be specified before the char_library command.

Example

Characterize with a user-define waveform

```
set_var normalized_driver_waveform { \
    0    0    \
    1.00E-12    0.0001    \
    1.00E-11    0.001    \
    2.00E-11    0.01    \
    3.90E-11    0.05    \
```

```
    5.20E-11    0.1    \
    6.00E-11    0.15    \
    6.50E-11    0.2    \
    7.00E-11    0.25    \
    7.50E-11    0.3    \
    8.00E-11    0.35    \
    8.50E-11    0.4    \
    9.00E-11    0.45    \
    9.50E-11    0.5    \
    1.00E-10    0.55    \
    1.05E-10    0.6    \
    1.10E-10    0.65    \
    1.15E-10    0.7    \
    1.20E-10    0.75    \
    1.25E-10    0.8    \
    1.30E-10    0.85    \
    1.36E-10    0.9    \
    1.46E-10    0.95    \
    1.55E-10    0.97    \
    1.65E-10    0.98    \
    1.85E-10    0.99    \
    2.05E-10    0.999    \
    2.25E-10    0.9999    \
    2.50E-10    1    \
}
```

non_seq_copy_dst_pin

`<pin_name>`	Name of pin for copy/transpose operation

These two parameters work together, and instruct Liberate to copy and transpose the following:

- **non_seq_setup** values under copy-source pin into corresponding **non_seq_hold** values under copy-destination pin
- **non_seq_hold** values under copy-source pin into corresponding **non_seq_setup** values under copy-destination pin.

This parameter must be specified before the `write_library` command.

Example

```
set_var non_seq_copy_src_pin CDN
set_var non_seq_copy_dst_pin SDN
```

non_seq_copy_src_pin

`<pin_name>`	Name of pin for copy/transpose operation

non_seq_pin_swap

`<0 | 1>`	Controls swapping of the constrained pin and related_pin when characterizing `nonseq_setup/hold`. Default: `1`
`0`	Liberate swaps the pin and the related_pin.
`1`	Liberate does *not* swap the pin and related_pin.

This parameter must be used before the `char_library` command.

nonseq_as_recrem

`<0 | 1>`	Controls whether the arcs with timing-type of `nonseq_setup/hold` should be converted into `recovery/removal`. Default: `0`
`0`	Arcs with timing-type of `nonseq_setup/hold` are not converted.
`1`	All arcs with the timing-type of `nonseq_setup/hold` are converted into `recovery/removal`.

This parameter can be used after the `char_library` command.

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 the internal probe pins should be included in the `.lib` file.
Default: `0`

- `0` Do not output the internal pins into the `.lib` file.
- `1` Output all internal probe pins into the `.lib` file with the `internal_pin` type.

This parameter must be used before the `char_library` command.

packet_arc_job_manager

`<string>` Enables the traditional or the Bolt job distribution system.
Default: `traditional`

Valid values: `bolt` and `traditional`

See also Bolt Job Distribution System and Distributed Processing

packet_arc_notification_interval

`<value>` Specifies the minimum time interval between two informational notifications (see `packet_arc_notification_list`). The range is between 0 to 72000.
Default: `600` (in Seconds = 10 minutes)

This parameter must be used before the `char_library` 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 parameter is effective when the `packet_arc_notification_list` has been set. The range is between 0 to 100.
Default: `10`

This parameter must be used before the char_library command is run.

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 parameter to a valid e-mail address to enable notifications to that address.
Default: `""` (empty list)

Requirements:

- The main Liberate 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 parameter must be used before the char_library command is run.

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_arc_optimize_idle_clients

`<0 | 1>` Controls whether the number of idle clients should be reduced.
Default: `0`

`0`	Does not reduce the number of idle client.
`1`	Reduces the number of idle clients. This could however increase wall time if there is a large variation in CPU time required to characterize each cell. This parameter is best used when CPU time required to characterize all cells are similar.

This parameter must be used before the char_library command is run.

Example

```
set_var packet_arc_optimized_idle_clients true
```

packet_arc_write_library_only

`<0 | 1>` Controls distribution of the write_library jobs to the clients based on the rsh_cmd parameter setting.
Default: `0` (disabled)

`0`	Disables distribution of the `write_library` jobs to the clients.

1	Enables distribution of the `write_library` jobs to the clients based on the `rsh_cmd` parameter setting whenever a distribute library command is encountered for the first time. While distributing the jobs to the clients, individual cell libraries are created for all the `write_library` commands. Then, all the individual libraries are appended on the server. For the subsequent `write_library` command runs on the server, no distribution happens when individual libraries are appended. **Note:** To run this mode, you do not need netlist or templates.

This parameter must be specified before the read_ldb or write_library command is run.

packet_arcs_per_thread

`<number>`	Sets the number of arcs each thread will characterize during arc-based distribution. Default: `10` (Each thread characterizes 10 arcs)

In the arc-based flow, Liberate can distribute the arcs of a cell across multiple machines. This parameter controls how many arcs are simulated by each thread. Larger values mean more arcs will be run on each thread, reducing the preprocessing and distributed overhead but increasing the runtime of that client. Cells with short preprocessing time but long simulation time will benefit from a smaller value for `packet_arcs_per_thread`, while cells with a longer preprocessing time and shorter simulation time benefit from larger values.

The default and recommended setting is `10`. For a large number of standard or base cells (that is, cells with mixed simulation times), a performance gain can be seen by using a setting of `40`.

For a small number of cells with longer simulation time (for example, block or multibit), a setting of `3` will yield greater distribution of jobs across clients.

This parameter must be used before the char_library command is run.

packet_arcs_per_thread_auto_adjust

`<0 | 1>` Adjusts automatically the packet_arcs_per_thread to fit the number of `packet_clients` and the specified threads.
Default: `0` (off)

- `0` Honors the user-provided setting for `packet_arcs_per_thread`.
- `1` Adjusts `packet_arcs_per_thread` based on packet_clients and `-thread` settings.

Setting this parameter causes `packet_arcs_per_thread` to automatically be increased (if necessary) so that all arcs can be distributed into a minimum number of jobs. This can help improve performance in cases where the pre-processing time comprises a significant amount of run time, or one cell is run by itself. The downside is that jobs with a few extremely long arcs might get bundled together; thus, lengthening the wall time of the entire run.

This parameter must be specified before the char_library command is run.

packet_cell_max_fets

`<value>` Controls the threshold below which the packet arc flow does not splits a cell.
Default: `25`

This parameter is only used when packet_mode is "arc". For example, if the value of packet_cell_max_fets is set to 100 then all cells with less than 100 devices will not be divided into smaller work units by the arc packet flow.

Minimum value allowed: 0 (all cells are divided)
Maximum value allowed: infinity (no cells are divided)

This parameter must be specified before the char_library command is run.

packet_client_idle_count

<number>	Specifies the allowed number of idle clients. Default: `-1` (Number controlled by internal heuristics)

Set this parameter to reduce the number of clients available for characterization near the end of the run. The default and recommended value is -1, which allows the tool to internally control the client release mechanism.

While characterizing statistical delay, setting this parameter to `2` helps to release idle clients in a more timely manner.

This parameter must be used before the char_library command is run.

packet_client_resubmit_count

<number>	Specifies the number of times a failed LSF job should be resubmitted. Default: `0`

This parameter specifies the number of times a failed LSF job should be resubmitted for simulation.

Note: Liberate will also check the LDB to make sure it contains data from the job, and will resubmit if necessary.

This parameter must be used before the char_library command is run.

packet_client_timeout

<value>	Sets a timeout value in seconds for client machines on the network. The valid values are 3600 (1 hour) to 86400 (1 day). Default: `86400 seconds` (1 day)

This parameter 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 cannot be trapped.) If a packet client is determined to be “dead”, then the server will not wait and move on to the next client.

This parameter must be used before the char_library command is run.

packet_client_timeout_action

`<error | warning>` Timeout action is not triggered unless the `packet_client_timeout` parameter is set to a non-negative value. The default action is `warning`.

- `error` — The master marks the job as failed (and the associated cell). If using LSF, the client job is killed using `bkill`.
- `warning` — Output a warning into the log file when the `packet_client_timeout` is exceeded. The timer is reset and the warning is repeated again and again after the timeout is exceeded.

This parameter has no effect unless you set packet_client_timeout to a valid value.

This parameter must be used before the char_library 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` (Packet-mode off)

- `0` — Parallel Packet mode off. (Default)
- `<integer>` — Enables Parallel Packet Mode *and* sets the number of machines to be used.

Note: If your flow uses write_vdb, you must set this parameter to `0`. If you are using the multi-PVT (both M1/nPVT) characterization flow of Liberate Trio, ensure that the `packet_clients` parameter is ***not*** set to `0`.

This parameter must be used before the char_library command is run.

packet_log_filename

`<file name>`	Name of log file. Default: `"log"`

Must set this to match the log file specified in the `rsh_cmd` to report characterization statistics. We recommend using the default name `"log"` and also setting `rsh_cmd` to use `"/log"` as stdout and stderr filenames.

Note: "%L" is not allowed as a string in `packet_log_filename`.

packet_mode

`<cell | arc>`	Controls the parallel packet distribution mode. Default: `arc` Liberate can distribute parallel packets in cell-based mode or arc-based mode. `cell` Enables cell-based mode. `arc` Enables arc-based mode.

This parameter must be used before the char_library command is run.

packet_require_spectre_char_opt

`<0 | 1>` Defines how the packet flows (see packet_mode) handle the `Spectre_char_opt` licenses.
Default: `1` (Use the `Spectre_char_opt` license if it exists)

- `0` Observes the setting of the spectre_use_char_opt_license parameter. When `spectre_use_char_opt_license` is enabled, Liberate can use the `Spectre_char_opt` licenses if they are available. If unavailable, Spectre checks out the MMSIM licenses. The `packet_client` continues even if it fails to checkout the required number of `Spectre_char_opt` licenses, relying on Spectre to manage the simulation licenses to be used for characterization. This provides the most flexibility in terms of license utilization at the risk of stability problems for large installations.
- `1` Uses only `Spectre_char_opt` licenses. This setting requires that `spectre_use_char_opt_license` is enabled. The `packet_client` waits until it checks out the required number of `Spectre_char_opt` licenses, with one license used per thread for normal jobs and two licenses used per thread for electromigration jobs. This setting is recommended if enough `Spectre_char_opt` licenses are available for the characterization run and/or the installation has more than 500 clients.

When starting a `packet_client`, Liberate has the option of acquiring and holding all licenses needed for characterization at once by checking out both client and `Spectre_char_opt` licenses. One client license is checked out per thread. If `Spectre_char_opt` licenses are not checked out for characterization, Spectre acquires licenses each time the Spectre process starts, allowing for utilization of any valid Spectre license (`Virtuoso_Spectre`, `Virtuoso_Multi_mode_Simulation`, `Virtuoso_Acceler_Parallel_sc`), as required.

The flexibility in using any available Spectre license might have an impact on license server stability in large installations. For SKI runs, there are one or more checkout events over the course of the job. For Spectre runs, there is one check-out event for each simulation. FlexLM license servers can only handle between `1000` and `2000` simultaneous accesses per second. If the installation has more than `500` client licenses, then there is a risk of overwhelming the license server if `Spectre_char_opt` licenses are not used.

This parameter must be used before the char_library command is run.

packet_rsh_mode

`<lsf | nc | custom>` Instructs the server during the arc packet flow (see packet_mode) to automatically kill client jobs if the server job is interrupted.
Default: `custom`

This parameter is automatically set for the `bsub` and `nc` batch submission commands (see rsh_cmd and set_rsh_cmd).

This parameter may need to be explicitly set to the correct value if the rsh_cmd is pointing to a wrapper script instead of using the commands mentioned above.

Important

If the `nc` command needs to be run when the Bolt job distribution system is in use, ensure that the `packet_rsh_mode` parameter is set to `nc`. This is necessary for Bolt to work properly specially when the jobs need to be killed. For information about Bolt, see Bolt Job Distribution System.

This parameter must be used before the char_library command is run.

parenthesize_not

`<0 | 1>` Puts parentheses around parameter names that are negated using the exclamation mark (!).
Default: `0`

- `0` Removes parentheses from the parameter names, that is, !A
- `1` Puts parentheses around a parameter, that is, !(A)

Liberate treats the *not* expression as represented with parentheses around the parameter name in logic functions. For example, !(A)

This parameter must be used before the char_library command is run.

Example

```
# Remove () from parameter names
set_var parenthesize_not 0
```

parenthesize_sdf_cond

`<0 | 1>` Puts parentheses around all sdf_cond statement in output library.
Default: `0`

- `0` Ignores to put parenthesis around `sdf_cond` statements.
- `1` Puts parentheses around `sdf_cond` statements.

This parameter must be used after the char_library command.

parse_auto_define_leafcell

`<0 | 1 | 2>`

Controls whether Liberate should recognize leaf cells when parsing a netlist, if yes, then how should it happen.
Default: `1`

`0` Does not generate `define_leafcell` commands.

Note: This setting is provided for backward compatibility to LIBERATE 14.1 ISR3 and prior releases.

`1` Recognizes leaf cells automatically and prints `define_leafcell` commands into a log file. All of the user-provided `define_leafcell` commands are preserved and honored.

`2` Detects leaf cells automatically, but does not apply the detected leaf cells to the analysis. Print `define_leafcell` commands into a log file so that you can examine and fill in the missing information. For example: l, w, nfin, cjsw, and so on.

A leaf cell is the lower most instance that is found when flattening a netlist that is outside the model file. The define_leafcell command can be used to specify the leaf cells manually. The auto-detection of leaf cells is supported only when using the Spectre Front End (SFE) parser (see read_spice `-parser sfe`) and the extsim_model_include parameter is set.

This parameter must be used before the char_library command.

parse_filter_rcs_mode

`<0 | 1>`

Controls whether reduction should be applied for RC circuits during characterization.
Default: `0`

`0` Do not apply the reduction.

`1` Apply the reduction for RC circuits.

This parameter must be used before the char_library command.

parse_ignore_duplicate_subckt

`<0 | 1>` Controls the behavior of Liberate when a duplicate subcircuits is encountered during characterization.
Default: `0`

- `0` Generates an error when a duplicate subcircuits is found.
- `1` Ignores all duplicate subcircuits and uses the first definition of such subcircuits.

This parameter must be used before the char_library command.

parse_remove_floating_fets

`<0 | 1>` Controls whether transistors that are gate-connected to floating nets should be removed.
Default: `0`

- `0` Do not remove the transistors that are gate-connected to floating nets.
- `1` Removes the transistors that are gate-connected to floating nets.

To remove floating SPICE models, you need to run only this parameter. There is no need to set the `define_leafcell -type black_box` command or any other parameter.

This parameter must be used before the char_library command.

parse_spectre_use_parhier_local

`<0 | 1>` Specifies the parameter passing rules to apply when using the read_library `-format spectre` command.
Default: `0`

- `0` Follows the setting of `.option parhier=local/global` in netlist. If this setting is not specified in the netlist, the `parse_spectre_use_parhier_local` parameter defaults to the global parameter passing rules.
- `1` Applies the local parameter-passing rules when using the read_spice `-format spectre` command.

This parameter must be used before the `read_spice` command.

pin_based_leakage

`<0 | 1>` Controls whether or not to output pin based `leakage_power` groups.
Default: `1`

- `0` Outputs only one leakage_power per state condition.
- `1` Outputs leakage_power groups based on state and power.

This parameter must be used before the char_library command.

pin_based_power

`<0 | 1 | 2>` Controls how pin-based power is calculated and characterized.
Default: `1`

- `0` Models only the power in the positive supply nodes (see `set_vdd`). This setting is not compatible with advanced power formats such as CCSP and ECSMP.
- `1` Models the power for all of the power pins of the cell (see `set_vdd` and `set_gnd`).

`2` This is similar to mode 1 with the addition that Liberate monitors the Miller capacitance current to each cell input. If the input is set to a positive or negative voltage, that current is added to the vdd or gnd power. Typically, this Miller capacitance current is not accounted for in any other power measurement. This is because when the driving cell is characterized, the receiving cell is not included, and even if it is, it may not be switching in the correct way to trigger the Miller capacitance currents. See the Performing Characterization using Liberate section for information on power characterization.

The `pin_based_power` parameter when set to `0` can co-exist with `voltage_map 1`. When this specific condition is encountered, the positive supply power will have CV^2 subtracted. When this parameter is greater than 0, both positive and negative supplies will have $\frac{1}{2}CV^2$ subtracted from them, depending on the value of `power_subtract_output_load`. The VSS power will not be output when using `pin_based_power` 0 in NLDM format libraries.

This parameter must be used before the char_library command.

Example

```
# Set the power calculation to monitor only VDD power
set_var pin_based_power 0
```

pin_based_signal_level_mode

`{list}` Determines when to write out the `input_signal_level` and `output_signal_level` attributes into the output library. Default: `0`

The following values are supported in the list: `""`, `0`, `1`, `input`, `output`, `inout`, and `all`.

`""` Same as the setting of `0`.

`0`	Includes only the `related_power_pin`/ `related_ground_pin` or `input_signal_level`/ `output_signal_level` attribute. **Note:** This setting is for backward compatibility with Liberate 18.1 ISR1 and prior releases. It is the same as using a setting of `""`.
`1`	Includes both `related_power_pin`/ `related_ground_pin` and `input_signal_level`/ `output_signal_level` attributes. **Note:** This setting is for backward compatibility with the Liberate 18.1 ISR1 and prior releases. When voltage_map=`1`, include both types of attributes. This is the same as using the setting of `all`.
`input`	Models only `input_signal_level` for input pins.
`output`	Models only `output_signal_level` for output pins.
`inout`	Models both `input_signal_level` and `output_signal_level` for inout (Bidi) pins.
`all`	Models `input_signal_level` and `output_signal_level` for all pins.

Note: If a supply referenced by a `*_signal_level` attributes is not included in a `pg_pin`, a `pg_pin` will be added to the output library.

This parameter must be used before the write_library command.

Example

```
set_var pin_based_signal_level_mode "input output"
```

pin_capacitance_matching_mode

<0 | 1>	Set pin_capacitance_matching_mode to 1 for the behavior of version 3.0p2 (or older) where set_pin_capacitance expects a complete match between the characterized when conditions and the specified condition while considering capacitances. When this parameter is set to 0, the overlapping when states are also considered. Default: 0

This parameter must be used before the write_library command.

pin_level_attributes

<attribute_name>	Maintains the specified attributes at the pin level and restricts them from being promoted to the bundle level. Default: "" (an empty list) This parameter appends to a predefined list of "input_map internal_node state_function". In addition, all "*_capacitance" and "*_transition" attributes are always kept at the pin level. **Note:** You can specify a whitespace separated list (space and tabs) of attributes with this parameter.

This parameter must be used before the write_library command.

Example

```
set_var pin_level_attributes "direction"
```

Forces the direction attribute to remain under each pin instead of appearing once at the bundle level.

Before:

```
bundle (D) {
    direction : input;
    members (D1, D2);
    pin (D1) {
        max_transition : 0.56;
        capacitance : 0.000555388;
```

```
    }
    pin (D2) {
        max_transition : 0.56;
        capacitance : 0.000555388;
    }
}
```

After:

```
bundle (D) {
    members (D1, D2);
    pin (D1) {
        direction : input;
        max_transition : 0.56;
        capacitance : 0.000555388;
    }
    pin (D2) {
        direction : input;
        max_transition : 0.56;
        capacitance : 0.000555388;
    }
}
```

pin_type_order

`{ pin_order_list }` Specifies the grouping of pins for a cell when written out to the library.
Default: `internals inouts outputs inputs`

The pins in a cell may be sorted into groups based on pin type as they are written out to the library. For example, all output pins may be put in one group and all input pins may be put in another group.

When pin sorting is enabled, this parameter sets the pin group order to be applied while writing the pins to the library; for example, it sorts input and output pins and places them in two separate groups. To enable pin sorting, the parameter sort_pins must be enabled. The currently supported list of keywords include:

- internals - All internal pins
- inouts - All bidirectional pins
- outputs - All output pins

- inputs - All input pins

Set this parameter to "" (the empty string) to request that if pin sorting is enabled (see sort_pins) then all pins are sorted alphanumerically.

This parameter must be used before the write_library command.

Example

```
set_var sort_pins 1
set_var pin_type_order {internals inouts outputs inputs}
```

pin_vdd_supply_style

`<0 | 1 | 2>`

Controls how the pin supply is reported in the Liberty model. The `vddName` are specified in the set_pin_vdd and/or set_pin_gnd `supply_name` options.
Default: `0` (Do not output a `vddName`)

`0` Do not output `vddName`.

`1` Behavior depends on setting of `voltage_map`:

`voltage_map=0`: Use `related_power_pin`
`voltage_map=1`: Use `input/output_signal_level`

`2` Output `vddName` in the .lib files in the format as specified by `voltage_map`.

This parameter must be used before the char_library command.

power_add_input_pin

`<0 | 1>` Specifies whether to include non-switching side input pin current.
Default: `1` (include input pin current)

- `0` Ignores the power contribution from non-switching side input pins regardless of the **pin_based_power** setting.
- `1` Liberate adds non-switching side input pin power according to the **pin_based_power** settings of `1` or `2`.

This parameter must be used before the char_library command.

power_adjust_for_pin_load

`<0 | 1>` Controls whether the harness power should be removed from the `internal_power` cell.
Default: `0`

- `0` Does not separates the power.
- `1` Separates the power consumed by the pin_load in the harness from the internal_power cell.

This parameter must be used before the char_library command.

power_binate_arc

`<"separate" | "merge">`

Controls whether the power should be split into multiple groups based on a WHEN condition. This parameter is used when multiple timing tables exist based on the timing_sense.
Default: `"merge"`

- `"separate"` Creates state-dependent power tables.
- `"merge"` Merges power into a single table that is not state dependent.

This parameter must be used before the char_library command.

power_combinational_include_output

`<0 | 1>` Controls whether the output pin is included in combinational cell leakage and hidden power `when` conditions.
Default: `1`

- `0` Excludes the output pin from the `when` state.
- `1` Includes output pin in leakage `when` states.

This parameter must be used before the char_library command.

power_divide_num_switching_mode

`<0 | 1 | 2>` Specifies that the power is divided by the number of outputs and inputs switching simultaneously. Simultaneous switching inputs are enabled using `define_arc -vector` and/or `simultaneous_switch_from_cell_when` and/or `simultaneous_switch`. Simultaneous switching outputs are controlled by the design itself, but can be indicated using the `define_arc -vector` option.
Default: `0`

- `0` The hidden power (input pin power) is divided by the number of switching inputs, and the switching power (output pin power) is divided by the number of switching outputs. This is the default and recommended value.
- `1` The hidden power is divided by the number of switching inputs, and the switching power is divided by the number of switching inputs after dividing by the number of switching outputs.

2	Liberate reports the full power (no division) on each pin and is used solely to verify the measured power without any processing. It may be desirable to also use power_subtract_hidden=0 and power_subtract_leakage=0 in order to start with the raw power numbers and then introduce each post-process in turn. This setting is used for debugging reported power.

This parameter must be used before the char_library command.

power_info

<0 | 1 | 2>	Print additional messages regarding power calculations. Default: 0 (no extra information)
0	Print no extra information. (Default and recommended for production flows)
1	Add calculations for the first point in each power table.
2	Add calculations for all points in every power table.

Validating power calculations is one of the challenging aspects of library qualification, especially since the relevant power data may not be contained in a single SPICE simulation. Enabling this feature outputs the additional information that may be used for power debugging or validation purposes.

For each power table, the additional data consists of the following:

- Cell
- Arc
- When
- Vector space covered by the arc and vector selected by Liberate
- Deck location
- Leakage power state selected
- Hidden power state selected
- Equation used

Some of the power calculations are done prior to `write_ldb`, while other calculations are performed after `read_ldb` or `write_ldb` but before `write_library`. Therefore, it might be necessary to consider both the options in order to arrive at the final power value. The separation should help to improve clarity and understanding of how the final numbers are generated. See power_info_filename for additional information.

Note: This feature is not supported for distributed jobs. It is only meant to be run locally for debug purposes.

This parameter must be used before the char_library command.

power_info_filename

`<file_name>`	Name of log file. Default: `powerInfo.log`.

If power_info is greater than 0, then the information is written to the log files corresponding to this parameter setting. Because power is calculated at two places, two log files are generated:

- **powerInfo1.log**: Contains Spice to LDB equations (leakage and load power).
- **powerInfo2.log**: Contains LDB to Library equations (hidden power).

These logfiles are saved under the `extsim_deck_dir` directory unless a full path is given. The name given in this parameter is changed to separate logs for each step.

This parameter must be used before the char_library command for all calculations.

power_minimize_switching

`<0 | 1>`	Enables an alternative algorithm for power vector selection. This is only needed for multi-output cells where the outputs do not always switch together. For example: multi-bit seq, some single-bit seq with async control, and multi-bit comb. Once enabled, this is not automatically disabled for single-output cells. Default: `0` `0` Delay and power vectors are the same.

1	Delay and power vector selection are decoupled. The power vectors are selected so that the minimum number of outputs are switching. This gives the most reliable total power when the internal behavior and the behavior at each switching output are combined by power analysis tools. This increases the total number of simulations (original delay vectors + switching power vectors), but does not affect constraints or hidden power. While the number of simulations increases, it does not double. This is because the setting restricts power to vectors with the minimum number of switching outputs. This setting is only required for multi-output cells where the outputs do not always switch together (multi-bit seq, some single-bit seq with async control, multi-bit comb).

This parameter must be used before the char_library command.

power_model_gnd_waveform_data_mode

<0 | 1>

Controls modeling of `leakage_power` and `internal_power` for rails having 0v.
Default: `0`

0	Allows modeling of `leakage_power` and `internal_power` for rails having 0v.
1	Disables modeling of `leakage_power` and `internal_power` for rails having 0v.

This parameter must be used before the write_library command.

power_multi_output_binning_mode

`<0 | 1>` Controls binning methodology for cells with multiple output pins.
Default: `0`
Recommended: `1`

For cells with multiple outputs, it is possible that the switching condition of one output might be completely unrelated to the switching condition of another output. For example, one output pin might rise while another could rise, fall, or not change. We recommend that binning take into account timing and power, especially where hidden power and leakage power are subtracted from separate decks.

`0`	Bin for timing effects only
`1`	Bin for both timing and power effects

This parameter must be used before the char_library command.

power_sequential_include_complementary_output

`<0 | 1>` Specifies whether to include a complementary output pin is included in sequential cell leakage and hidden power `when` conditions.
Default: `1` (Include output)

`0`	Excludes the complementary output pin from the `when` state.
`1`	Includes output pin in leakage `when` states.

This parameter must be used before the write_library command.

power_settings_reduce

`<0 | 1>` Specifies whether to retain all power settings for a cell or only the first unique ones.
Default: `1`

`0`	Retain all power settings for each cell.
`1`	Retain only the first power settings that are unique in the supply levels for a cell.

Note: This parameter can be set on a per-cell basis.

This parameter must be used before the char_library command.

power_settings_when

`<mapping>` Specifies a mapping of power setting names to when conditions. This is useful to define any input settings that must be active to apply a particular power setting.

The power setting names are defined using the add_power_setting command.

Note: This parameter can be set on a per-cell basis.

Example

```
set_var -cells RETFF power_settings_when { off1 !RETN off2 RET }
```

power_sim_estimate_duration

`<0 | 1 | 2>` Allows the user to specify a different method for the power tend.
Default: `2`

`0`	Not implemented.
`1`	Estimates the power tend using the slew measurement threshold.
`1`	Used in LIBERATE 12.1 ISR3 and earlier behavior. This setting is for backward compatibility.

This parameter follows the `tran_tend_estimation` mode for consistency.

power_subtract_leakage

`<0 | 1 | 2 | 3 | 4 | 6 | 7>`

Determines what method is used to subtract leakage power from the energy numbers in the `internal_power` tables. (See Leakage Power for more information.).
Default: `1`

`0` No leakage subtract is done.

`1` Leakage is determined from the separately computed leakage values found in the .lib. The leakage subtracted is the average of the leakage matching the initial (pre-transition) and final (post-transition) states.

Note: With this setting, the leakage is only subtracted from internal_power tables when voltage_map=0 and pin_based_power=0. (Default)

`2` Leakage is determined from the final (post-transition) state current as measured in the energy measurement deck. The leakage will be subtracted from all internal_power tables (All combinations of voltage_map and pin_based_power).

`3` Leakage is computed as in mode 1, but is subtracted from all internal_power tables (All combinations of voltage_map and pin_based_power).

`4` Leakage is determined from the final (post-transition) state current as measured in a separate "leakage" measurement deck. It is subtracted from all internal_power tables (All combinations of voltage_map and pin_based_power).

Note: An incomplete set of `leakage_groups` might lead to incorrect leakage subtraction. This option should not be used if the separate leakage simulations do not contain complete coverage for all input and output pin post-transition states simulated for a cell.

`6`	Determines the time (`Leakcross`) when the output crosses the delay measurement threshold. Liberate will subtract the initial state (pre-transition) leakage power up to `Leakcross` and the final state (post-transition) after `Leakcross` from the internal power tables. The initial and final leakage is measured within the same deck as where the power is measured.
`7`	This value follows the same method as the setting of `6`, but takes the pre-state and post-state leakage values from separate *leakage* measurement decks.

This parameter must be used before the char_library command.

power_subtract_leakage_msg_level

`<0 | 1>` Enable messages for state matching during leakage subtraction.
Default: `0`
Recommended: `1`

`0`	No warnings reported, even when mismatches occur.
`1`	Warnings for mismatched or missing leakage states reported.

This parameter enables an additional check when `power_subtract_leakage` is enabled. If the leakage state matching the pre- or post-transition criteria is not an exact match, Liberate reports which state was used for subtraction.

As `power_subtract_leakage=2` is the only method where in-deck leakage values are used, settings 1, 3, and 4 can have a partially-expanded set of leakage states, which would result in subtraction of the default leakage power. It is recommended to enable this check if full leakage state coverage is expected as warnings indicate missing states.

This parameter must be used before the char_library command.

power_subtract_leakage_mode

`<0 | 1 | 2>`

Controls the method Liberate uses to subtract leakage power from `internal_power` tables.
Default and Recommended: `2`

- `0` — Behavior in 2.5p2 and prior releases. Multiple outputs and WHEN states are not considered.
- `1` — Switching power is summed for cells with multiple outputs, leakage is subtracted, and then the remainder is divided by the number of switching output pins.
- `2` — Behavior in 1 plus consideration for the leakage WHEN state in hidden power calculations for table-based default internal_power groups. Does not apply to bitwise default groups.

This parameter must be used before the char_library command.

Example

```
# Apply power_subtract_leakage_mode=2 properly.
set_var power_subtract_leakage 3
set_var power_subtract_leakage_mode 2
set_default_group -criteria {power max leakage min} -method {default table}
```

power_subtract_output_load

`<none | all | one | one_rise_full | one_both_half | all_rise_full | all_both_half>`

Specifies the method to be used to subtract the output load from the internal power.
Default: `all` (Subtract output load energy for all switching outputs from internal power.)

- `none` — Do not subtract output load power.
- `all` — If pin_based_power=0, subtract CV^2 for all rising outputs.
  If pin_based_power=1, subtract 0.5CV^2 for all rising or falling outputs.

`one` If pin_based_power=0, subtract CV^2 for the rising pin.
If pin_based_power=1, subtract 0.5CV^2 for the rising or falling pin.

`one_rise_full`

Subtract CV^2 for the rising pin.

`one_both_half`

Subtract 0.5CV^2 for the rising or falling pin.

`all_rise_full`

Subtract CV^2 for all rising outputs.

`all_both_half`

Subtract 0.5CV^2 for all rising or falling outputs

Note: When all, all_rise_full, or all_both_half are selected, then the internal power for all arcs originating at the switching related_pin will be divided by the total number of output pins. This assumes the power tool will add the internal power from all of the arcs of the related_pin. For example, a DFF with one clock and 2 outputs will have 2 arcs in the library, one from clock to Q and one from clock to QB. The internal power from a single clock transition will be divided between the 2 outputs.

This parameter must be used before the char_library command.

power_subtract_output_load_mode

`<0 | 1 | 2 | 3>` Specifies the formula used to compute the power consumed in the output load. The resultant power is subtracted from the switching `internal_power`.
Default and recommended: `3`

Note: When computing the switching `internal_power`, Liberate subtracts the power used to drive the output load. Liberate will subtract CVV or 0.5CVV depending on the setting of the pin_based_power and power_subtract_output_load parameters. The `C` in CVV is the output load capacitance. `Vsupply` is the supply rail voltage swing (`Vvdd-Vvss`). `Vfinal` is the final voltage on the output pin.

`0` When using Alspice, subtract:
`0.5*Cload*Vsupply*Vfinal`

When using extsim, subtract:
`0.5*C*Vfinal*Vfinal`

The `0.5` factor above is determined by the value of the pin_based_power parameter.

`1` Subtract `0.5*C*Vsupply*Vsupply`.

`2` Subtract `0.5*C*Vsupply*Vfinal`. (All simulators)

3 Subtract `0.5*C*Vsupply*Vfinal`. This setting modifies the behavior in the setting of `2` where if `ecsm_measure_output_range=1` and `power_minimize_switching=0`, the `Vfinal` was replaced with `Vsupply`.

Use this parameter to determine the voltage value represented by `Vfinal` when subtracting the output load related power.

This parameter must be used before the char_library command is run.

power_tend_match_tran

`<0 | 1>`

Forces power_tend to match tran_tend in SPICE simulations when using external simulators (see char_library -extsim) without the SKI interface.
Default: `0`
Recommended: `1`

`0` Do not match power_tend to tran_tend (Default).

`1` Match power_tend to tran_tend.

When setting up SPICE decks, Liberate can measure power to the end of the transient simulation or to a prior point. In well-behaved circuits, the current should stabilize prior to the end of the simulation so that either of the settings gives the same result. However, if a circuit exhibits trapezoidal ringing, and the purpose is to match results from different simulators or methodologies (for example, correlate stand-alone Spectre with SKI); it is recommended to use matching measurement times.

Note: Trapezoidal ringing in the current waveforms lead to both inconsistent and incorrect power results. Setting this parameter only addresses the inconsistent part of the problem. This problem should be corrected by updating the extracted netlist or using different simulation options (gear or related integration methods).

When using the internal Alspice or integrated Spectre-SKI simulation engines, Liberate measures power by integrating the current to the end of the transient simulation (tran_tend). When using an external simulator, Liberate integrates the current to the estimated end of the transition (power_tend). There are cases where tran_tend or power_tend are different which can cause power correlation issues between Alspice, SKI runs, and external simulator runs. This parameter forces power_tend to be equal to tran_tend, which improves power correlation. However, trapezoidal ringing in the current waveforms can lead to both correlation problems and incorrect power results. Setting this parameter only addresses the correlation part of the problem. A more appropriate solution might be to update the extracted netlist or use different simulation options (gear or related integration methods).

This parameter must be used before the char_library command.

power_vector_selection_criteria

`<off | min | max | avg>`

Determines how to select vector(s) for which multiple are available for a power (hidden or switching) arc, and how to combine them. If worst_vector_selection_mode=0, vector selection is disabled. However, the `power_vector_selection_criteria` parameter still controls combination of the power results for each vector.
Default: `off`

`off`	Choose vectors based on `set_default_group` (former behavior).
`min`	Choose the vector(s) with minimum power.
`max`	Choose the vector(s) with maximum power.
`avg`	Choose the min and max power vector(s) and do a point-by-point average to combine.

This parameter must be used before the char_library command.

predriver_waveform

`<0 | 1 | 2>`

Set a PWL waveform as the input driver based on averaging a linear ramp and the equivalent exponential response from an RC network.
Default: `0` (Use a linear map as the input slew.)

`0`	Disabled.
`1`	The linear ramp used will be limited to the supply voltage rails. Over-rides any set_driver_cell commands.
`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.

This parameter enables using a PWL waveform which is generated by averaging a linear input ramp and a step response of an RC network as the pre-driver.

Use this analytical waveform to give 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.

Important

1. This parameter is disabled when the predriver_waveform_ratio parameter is set to `0`.
2. For library validation (LV) if a normalized wavewform exists in the library, it will override the predriver_waveform parameter setting.

While characterizing NLDM delay models, you can use this parameter for defining the input waveform shape. For detailed information, see Delay Models in Chapter 7, "Performing Characterization using Liberate."

This parameter must be used before the char_library command.

Example

```
# Use a PWL pre-driver derived from an RC network
set_var predriver_waveform 2
```

predriver_waveform_mode

`<0 | 1>` Controls the inclusion of the slew measurement thresholds in the predriver waveform.
Default: `1`

- `0` Use best curve fit result.
- `1` Include the exact slew measure threshold values in the input waveform. Use curve-fitting to determine the remaining points. This is also the recommended setting because it removes the interpolation errors of the input waveform from the delay measurements.

This parameter instructs Liberate to include the exact transition (see `measure_slew_*`) threshold values in the input waveform. The trade-off is that the accuracy of the best curve-fit waveform may be sacrificed in order to include the exact slew measure threshold values.

This parameter must be used before the char_library command.

predriver_waveform_npts

<value>	Set this parameter to the minimum number of points used to store each driver waveform. Default and Recommended: 17

This parameter must be used before the char_library command.

predriver_waveform_ratio

<value>	Specify a ratio of the linear to the exponential waveforms being merged. Default: 0.5 (50%)

The CCS predriver waveform is created from a summation of a linear ramp and an exponential waveform. Use this parameter to control the weight applied to the linear ramp waveform versus the exponential waveform. (Default value is 0.5, acceptable values are in 0.0 - 1.0). This parameter is works anytime the predriver_waveform is set to a non-zero (enabled). When predriver_waveform_ratio is set to 0, predriver_waveform is disabled.

v_weighted = v_linear * predriver_waveform_ratio +
v_exponential * (1-predriver_waveform_ratio);

This parameter must be used before the char_library command.

Example

```
# Use a PWL pre-driver waveform with a more linear behavior
set_var predriver_waveform_ratio 0.65
```

preserve_user_function

`<0 | 1>`

Used during Verilog/Vital modeling to preserve the function given previously instead of regenerating it from AND/OR/INV constructs.
Default and Recommended: `1` (preserve the function)

- `0` Generate the logic function internally. This is the behavior of pre-3.1 versions of Liberate. Under certain circumstances, this can introduce additional pessimism during X-propagation tests.
- `1` Do not regenerate the logic function. Instead, the function must come from user_data given during Liberty modeling, or a function read in during `read_ldb` or `read_library`. This is the default and recommended behavior as of release 3.1.

Liberate has the ability to generate a logic function from basic building blocks: AND, OR, INV. This is used during Verilog and Vital modeling.

prevector_period

`<value>`

Sets the period used by the prevector option of the `define_arc` command. Each vector listed in the prevector is simulated for the time period specified by this parameter.
Default:`1e-8 (10ns)`

This parameter must be used before the char_library command is run.

prevector_slew

`<value | "min" | "mid" | "max" | "index_n">`

Specifies the prevector slew rate to apply to the input and bidi pins in the `-prevector_pinlist` of the `define_arc` command.
Default: `0.1e-9 (100pS)`

- `value` Specify the exact slew to use, in seconds.
- `min` Uses the minimum slew from the slew index (usually `index_1`).

`mid`	Uses the middle slew from the slew index.
`max`	Uses the max (largest) slew from the slew index.
`index_n`	Uses the positional slew (beginning with position `0`) from the slew index.

This parameter must be used before the char_library command is run.

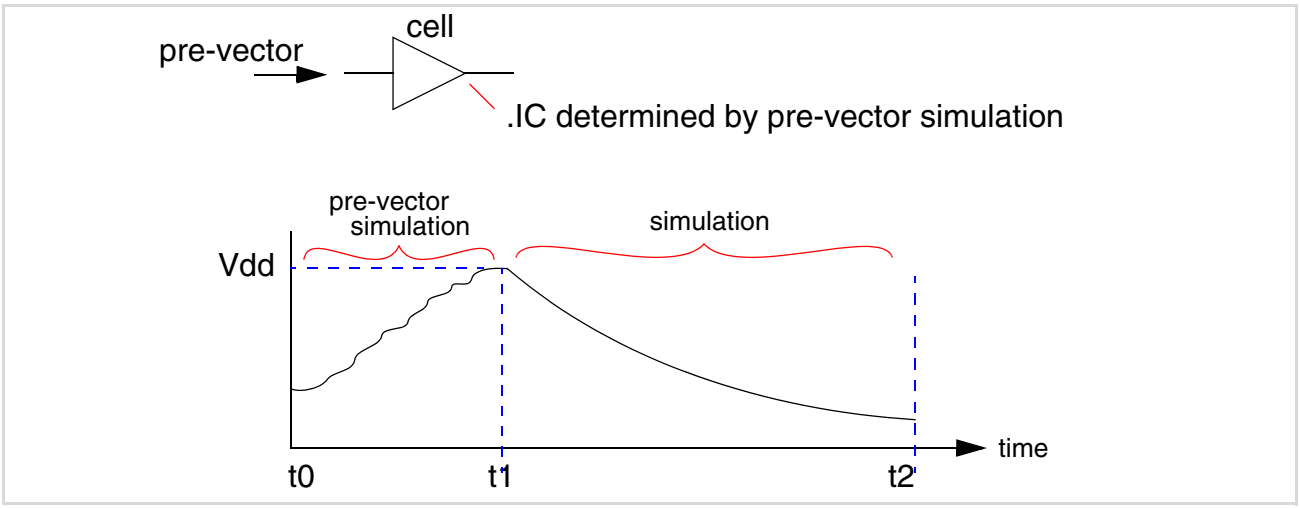
Example

```
index_0 = min = first slew
index_1 = second slew
index_2 - third slew
```

prevector_voltage_waveform_mode

`<0 | 1 | 2>` Selects which part of the pre-vector waveform will be stored in the Library.
Default and Recommended: `1`

`0`	Store waveform from **t0** to **t1**. ("Incorrect" behavior, prior to release 3.2)
`1`	Store waveform from **t1** to **t2**.
`2`	Store waveform from **t0** to **t2** . (Might be needed for some analog circuits.)



This parameter must be used before the char_library command is run.

process_match_pins_to_ports

`<0 | 1>` Set this to require strict port checking for the `define_cell` and `define_arc` commands.
Default: `0`

- `0` The `define_cell` command may have cell pins that are not in the subcircuit port list. If a `-pin` or `-related_pin` name of `define_arc` is not in the `define_cell` command for the specified cell, Liberate skips the `define_arc` command. The warning message is not printed out.
- `1` Enables strict pin to port mapping. Every pin in the `define_cell` command must map to a port in the subcircuit for the cell and every non-supply (see set_gnd) port must map to a pin in the `define_cell` command. If a pin does not map one to one with a port, Liberate outputs an error message and skip the cell for characterization.

Note: In a `read_library` followed by `write_template` flow, Liberate sets this parameter to 1 in the output template file to ensure that the `define_cell` pins and subcircuit ports match.

This parameter must be used before the char_library command is run.

process_node

<string> This parameter is typically set by the foundry to load custom settings for a certain process technology. Access to some reporting features might be restricted.
Default: `""`

This parameter must be used before the char_library command is run.

process_node_in_encrypted_file

<string> Specifies the path to encrypted file from which different node settings need to be enabled. Ensure that this parameter is set before you run the esource command.
Default: `""`

pwxsim_cap_mode

`<constrant | estimated>`

Controls how the internal capacitance load is determined for the switch-level simulator.
Default: `estimated`

`constant` Uses a fixed capacitance load for all wires.

`estimated` Uses estimates of the capacitance load from fanout transistors and channel connections.

For certain cells, such as, the large multi-bit cells, the estimated load on the clock pair can result in a large internal clock skew. If there is also an internal race condition, such as, between master/slave latch of a flip-flop, then some arcs might switch when they should not or conversely. This can be corrected using the `constant` mode to make the switch-level simulator insensitive to the clock fanout.

ramp_vsrc

`<0 | 1 | 2 |3>` Enables supply ramping.
Default: `0`

- `0` Disables ramping.
- `1` Ramps the power supply voltage sources depending on the `ramp_vsrc_mode` and `extsim_flatten_netlist` parameters.

  `ramp_vsrc_mode = 0, extsim_flatten_netlist !=0: ramp supplies and inputs`

  `ramp_vsrc_mode = 1, extsim_flatten_netlist !=0: ramp supplies, inputs, internal and outputs`

  `ramp_vsrc_mode = 0, extsim_flatten_netlist ==0: ramp inputs`

  `ramp_vsrc_mode = 1, extsim_flatten_netlist ==0: ramp inputs, internal and outputs`
- `2` Ramps the power supply voltage sources and inputs. The `ramp_vsrc_mode`, `extsim_model_include`, and `sim_use_init_duration` parameters have no effect.
- `3` Ramps the power supply, inputs, internal, and output voltage sources. The `ramp_vsrc_mode`, `extsim_model_include`, and `sim_use_init_duration` parameters have no effect.

Note: If `sim_use_init_duration` is set to `2` and `ramp_vsrc` is set to `0` or `1`, the same behavior is followed as if `ramp_vsrc` is set to `1`, including the impact of `ramp_vsrc_mode` and `extsim_model_include`.

This parameter must be used before the char_library command is run.

ramp_vsrc_mode

`<0 | 1>` Controls which cell ports should be ramped when `ramp_vsrc` is enabled.
Default: `1`

- `0` Ramps the power supplies and input ports.
- `1` Ramps the power supplies, input ports, internal nodes, and output ports.

This parameter must be used before the char_library command.

ramp_vsrc_ratio

`<value>` Specifies the ratio of the sim_init_duration parameter to use for ramping the supply.
Default: `0.5`

Note: This parameter works together with ramp_vsrc and `sim_init_duration`.

When `ramp_vsrc` is enabled and the `ramp_vsrc_ratio` parameter is set to `1`, Liberate initializes all voltage sources to `0V` at time zero, and then ramps those voltage sources to their intended voltage value at a time equal to `ramp_vsrc_ratio * leakage_sim_duration`.

This parameter must be used before the char_library command.

rc_floating_cap_mode

`<0 | 1 | 2 |4>` Controls the handling of floating-cap nodes.
Default: `0`

- `0` Disable automatic detection of floating nodes.
- `1` Adds a large dc-path resistor (fixed at 1e+14 ohms) from floating cap nodes to node 0.

2 Write out (and overwrite any existing) floating cap net names to a file named *<cell_name>*.ics in the current working directory. The user can examine and include this file to set .IC on floating nodes.

For Spectre, the file contents will look like this:

```
ic n1 = 0
ic n7 = 0
```

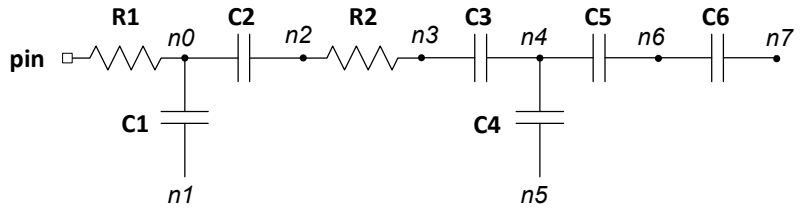
For other simulators:

```
.ic v(n1) = 0
.ic v(n7) = 0
```

4 Remove all floating caps and their direct connected RC network during simulation.

To assist with DC convergence and prevent a drastic increase in simulation run-time, Liberate can add terminating resistors to floating capacitor nodes. A floating node or network is one that is resistively connected: it has coupling capacitors to other nodes, but has no path (leakage or driving) to power or ground.

In the following circuit example, dc-path (1/gmin) resistors will be tied from node "0" to nodes: n1, n2, n4, n5, n6, n7 (n0 is not a floating node and n3 is current equivalent with n2.)



When this parameter is set to 4 and the example above is used, the result will be this:

This parameter must be used before the char_library command is run.

rcp_cmd

<scp | rcp | cp>	The file copy command to use for copying files from the host to the client machine when using distributed parallel processing. Default: scp

This parameter must be used before the char_library command is run.

rdb_checkpoint_dir

<directory_name>	Full path to the directory where the rdb checkpoint file is stored. Default: (there is no default)

If set, Liberate can store per-arc characterization data into a recovery database (RDB). This parameter is used to specify the directory where the RDB checkpoint files will be stored. This enables a per-arc recovery flow at the cost of additional drive space and runtime. By default, this data is not saved.

The rdb_checkpoint_dir parameter can be useful when characterizing extremely large IO cells that have very long run times. This is because Liberate will load the characterized arc data and proceed to characterize the missing arcs. However, there is no version control to prevent Liberate from mixing simulation data created using different settings.

We recommend *not* using this parameter.

This parameter must be used before the char_library command is run.

Example

```
set_var rdb_checkpoint_dir /NFS/work/rundir/RDB
```

rdb_delete_upon_completion

<0 | 1 | 2>	Controls whether the arc-level characterization database files for each cell should be retained or deleted. Default and recommended: 2
	0 — Do not delete. Retain the arc-level characterization database files for each cell.
	1 — Delete the arc-level characterization database files for each cell before the LDB is written.

2	Deletes the arc-level characterization database files for each cell after the LDB is written.

This parameter must be used before the read_ldb command is run.

During a characterization session, the tool generates a <ldbdir>/<pvt>/<cell>.rdb directory having the arc-level characterization database files for each cell. When the rdb_delete_upon_completion parameter is set to 1, these RDB files are automatically deleted after the tool has read them.

To enable arc-level recovery, these RDB files need to be retained. Therefore, ensure that the rdb_delete_upon_completion parameter is set to 0 to disable the automatic deletion of the RDB files. Preferably, set the parameter to its default and recommended value, 2, to ensure deletion of the RDB files only after the LDB has been written.

Note: In the Liberate Trio Multi-PVT (define_pvt-based) flow, the multi_pvt_recovery_flow parameter is set to 1 to enable the read_ldb-based recovery flow.

rdb_exit_if_source_differ

<0 | 1>	Exit if mismatched PVT corners are found. Default and recommended: 1

The RDB flow is enhanced to ensure that 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 parameter to the same location.

This parameter must be used before the char_library command is run.

Example:

```
# To enable this check:
set_var rdb_exit_if_source_differ 1
```

rechar_chksum

<string>	Name of the Linux file checksum utility. Must be in $PATH. Default: "" (Unset).

This parameter specifies the Linux utility that Liberate uses to generate a unique checksum for each netlist file read by Liberate. The checksum utility is used during the incremental recharacterization flow to determine if a new netlist has been loaded for a cell. The rechar_chksum utility is enabled in the incremental recharacterization flow when the read_ldb command is called with the -incremental option and is enabled by the read_ldb command arguments -incremental and -check_spice.

The recommended checksum utility is md5sum. There is no default; the user must set this.

This parameter must be used before the char_library command is run.

Example

```
set_var rechar_chksum md5sum
```

removal_glitch_peak

<value>	Glitch height as a ratio of supply used in characterizing removal constraints. Default: -1 (Use the same value as constraint_glitch_peak)

This parameter is used to specify the maximum size of a voltage glitch permitted on the constraint output pin before an arriving signal is deemed to fail a removal constraint. When this parameter is not set, then removal constraint characterization will follow the constraint_glitch_peak value.

This parameter must be used before the char_library command is run.

Example

```
# Set removal glitch peak to 5%
set_var removal_glitch_peak 0.05
```

res_merge

`<0 | 1>`	Allows Liberate to merge resistors to improve simulation run time. Default: `1` (on) Set to 0 to disable.

This parameter must be used before the char_library command is run.

res_open_tol

`<value>`	Controls to filter out (and leave the connection open) any resistor that is larger than this parameters setting, only if `res_open_tol` is set to a value larger than 0.

This parameter must be used before the char_library command is run.

res_tol

`<value>`	Threshold value to short resistors. Default: `0.0011`

Any resistors below this limit will be shorted (set to zero ohms), filtering out resistors that are suspected of being extraction artifacts.

A netlist containing very small resistors could produce an unstable simulation matrix, resulting in unnecessarily long simulations or inconsistent results. Therefore, it is important that the extractor is set up to generate netlists that are representative of the actual cell as it will be used in the design, and are well-behaved in SPICE simulation.

For certain extraction methodologies, it may be necessary to set this parameter to 0.1 for consistent results. Setting this to 0 will disable this functionality, *however*, the quality of the simulation results may deteriorate if the netlist contains the instabilities mentioned. If some simulations fail after changing this parameter, try setting minimum resistance control options in an external simulator to see if the results improve.

This parameter must be used before the char_library command is run.

reset_leakage_current_mode

`<0 | 1>` Controls when leakage will be reset to zero (`0`) amps.
Default and recommended: `0` (Zero negative leakage for Library Compiler compliance)

- 0 — Resets leakage current greater than `0` amps to `0` for `primary_ground` or `backup_ground` pg pins. Also, resets leakage current less than `0` amps to `0` for all other `pg` pins.
- 1 — Do not reset leakage current when the `pg_type` is `primary_power`, `primary_ground`, `backup_power`, or `backup_ground`.

This parameter must be used after the char_library command is run.

reset_negative_constraint

`<0 | 1>` Turn any negative constraint value to zero.
Default: `0`)

By default, Liberate will output a negative constraint (setup, hold, recovery, and removal) value if the simulator measurement returns a negative value. Setting `reset_negative_constraint` will cause the output library to have the negative values replaced by zeros. This reset will be applied after `set_constraint -margin`.

Note: This parameter will not be applied to mpw constraints.

This parameter must be used after the char_library command is run.

reset_negative_delay

`<0 | 1>` Turn any negative delay or transition values to zero.
Default: `0`

By default Liberate will output negative delay and transition values if the simulator measurement returns a negative value. Setting `reset_negative_delay` will cause the output library to have the negative values replaced by zeros.

This parameter can be used after the char_library command is run.

Example

```
# Turn all negative delays and constraints to zeros
set_var reset_negative_delay 0
set_var reset_negative_constraint 0
```

reset_negative_leakage_power

`<0 | 1>` Reset any negative leakage power sum to zero.
Default: `0`
Recommended: `1`

- `0` Do not reset negative leakage power.
- `1` Avoid using negative leakage currents in power calculation when the leakage has been zeroed out for LC compliance.

Note: When the reset_negative_power parameter is set to `1`, the `reset_negative_leakage_power` parameter has no effect and all negative power, including leakage will be reset.

This parameter must be used before the char_library command is run.

reset_negative_leakage_power_value

<value> Specifies the value (in Watts) that the negative leakage power value is reset to. The reasonable value for this parameter is zero or a small positive value at the leakage power level. This parameter is applied when the parameter reset_negative_leakage_power is enabled.
Default: `0.0`

Note: The value should be non-negative.

This parameter must be set before the write_library command is run.

reset_negative_path_delta_measurement

`<0 | 1>` Controls whether the negative measurement result of path delay is reset to `0` in the LDB.
Default: `0`

0	Retain the original value, do not reset it to 0.
1	Reset the value to 0, and print a warning message (LIB-478) in the log file.

If the -pin_probe and -related_probe options are used with the define_arc command, the delay measurements from pin to pin probe and the related pin to related probe will be reported in the LDB.

If the delay measurement is negative, use this parameter to control saving measurement result or 0 to LDB.

Note: These path delays in the LDB will be used only in margin calculations. For related details, see the -pin_probe_factor and -related_probe_factor options of set_constraint.

This parameter must be used before the char_library command is run.

Example

```
set_var reset_negative_path_delta_measurement 1
```

reset_negative_power

<0 | 1 | 2 | 3> Resets any negative power sum to zero.
Default: 1

0	Do not reset negative power; allow negative values for power.
1	If the sum of rising and falling power is negative, whichever power is negative will be adjusted so the sum of the rising and falling power is zero. If they are both negative, then both will be set to zero. See the algorithm shown in the description below. (Default)
2	Any power that is negative will be set to zero.

`3`	Resets the negative power only if the sum of all power rail `rise_power` and all power rail `fall_power` is less than `0`. When the sum is positive, do not reset any power values even if the sum for an individual rail is negative. When the sum is negative, adjust each rail so that the sum of rise_power and fall_power is positive for that rail.

By default, Liberate resets negative power so that the `rise_power + fall_power` should always be greater than or equal to zero.

Dynamic power entries are processed when "power pairs" are identified. (Power pairs have matching *when* conditions, same input/output pins, but different directions, that is, rise versus fall.) The sum of corresponding entries between rise/fall pairs is checked, and if it is less that zero (for example, R2+F2 < 0), then one of the following actions is performed:

```
( Rn < 0 ) & ( Fn < 0 ) then Rn = Fn = 0
( Rn < 0 ) & ( Fn > 0 ) then Rn = ( -1 * Fn )
( Rn > 0 ) & ( Fn < 0 ) then Fn = ( -1 * Rn )
```

If an arc does not have a matching rise/fall "power pair", then the above procedure is not performed.

Note: When characterizing CML logic, it may be necessary to disable the resetting of negative power. Also, negative *leakage_power* entries will be reset to 0.

This parameter must be used before the write_library command is run.

resolve_collision

`<0 | 1 | 2>`	Controls simulation engine used to resolve collision vectors with the requested spice engine. Default and Recommended: `2` (Use `-extsim` if specified)

`0`	Disables simulation-based collision resolution.
`1`	Use Alspice to resolve collisions.
`2`	Use the external simulator if `-extsim` is used in `char_library`.

Collision vectors occur when a node has a path to vdd and to gnd. In these cases, Liberate can simulate the vector in spice to determine if the output results in a digital behavior.

This parameter must be used before the char_library command.

retry_count

`<value>`	Specifies the number of retries for a particular arc, pausing 5 seconds between retries. Default: `1` (Instructs Liberate to retry failing simulations one time)

This parameter must be used before the char_library command.

retry_count_file_operation

`<value>`	Specifies how many times to retry when an attempt to move, copy, or rename a file fails due to network or LSF issues. A valid value is an integer between `0` and `60`, both inclusive. Default: `10`

This parameter must be used before the char_library command.

rsh_cmd

<ssh | rsh | string>

Controls the interface to remote clients when using distributed parallel processing. Before using parallel processing, make sure that the host machine (the machine from which Liberate is to be run) can perform the specified action.

When using `ssh` or `rsh`, the `rcp_cmd` parameter specifies the command that will retrieve the results from each client machine without requiring a password or passphrase. It is recommended to use a batch queuing system such as LSF. See Distributed Processing for more details about using the arc packet flow.
Default: `ssh`

This parameter must be used before the char_library command.

Example

```
# Set up Liberate to submit clients to an LSF batch queuing system.
set_var rsh_cmd "bsub -q myQueue -R "(OSNAME==Linux) \
     rusage\[mem=4000,swp=2000\]  span\[hosts=1\]" -P LIB:16.1:PE:RND \
     -W 10:0 -n 4 -o %B/log -e %B/log"
```

scalar_power_warning

`<0 | 1>`

Determines whether Liberate will output WARNING(LIB-308) when there is a scalar `rise_power` or `fall_power` group in the library. This parameter only applies to `internal_power` groups where the output pin is switching. Therefore, do not apply it to hidden power.
Default: `0`

- `0` — Disable WARNING(LIB-308) when there is a scalar `rise_power` or `fall_power` group in the library.
- `1` — Enable WARNING(LIB-308) when there is a scalar `rise_power` or `fall_power` group in the library.

This parameter must be used before the write_library command.

Example

```
set_var scalar_power_warning 1
WARNING (LIB-308): The data table for the arc of cell 'DFF2RX1', pin 'Q1',
'rise_power' is 'scalar'. If this is not expected, review the characterization log
file for possible causes such as failed simulations, update the Tcl script and
rerun.
WARNING (LIB-308): The data table for the arc of cell 'DFF2RX1', pin 'Q1N',
'fall_power' is 'scalar'. If this is not expected, review the characterization log
file for possible causes such as failed simulations, update the Tcl script and
rerun.
```

scale_load_by_template

<0 | 1>	Set this parameter to scale the load indexes using the values defined in the template when generating indexes using the -auto_index option of char_library. Default: 0 (off)

This parameter turns on scaling of the load indexes created by the -auto_index option of char_library. Normally, when this command option is enabled, the intermediate indexes are determined using a geometric sequence. When this control parameter is set, the indexes are determined using the following formula:

```
index_value(i)=(max_load-min_load)*
  template_load_index_value(i) + min_load
```

where:

i=2,…,number_of_indexes-1

template_load_index_value is given in a define_template command where the index values are treated as a ratio between 0 and 1. See scale_tran_by_template for an example.

This parameter must be used before the char_library command is run.

scale_tran_by_template

<0 | 1>	Set this parameter to scale the transition indexes using the values defined in the template when generating indexes using the -auto_index option of char_library. Default: 0 (off)

This parameter enables scaling of transition indexes created by the -auto_index option of char_library. Normally, when the -auto_index option of char_library is enabled, the intermediate indexes are determined using a geometric sequence. When this control parameter is set, the indexes are determined using the following formula:

index_value(i)=(max_tran-min_tran)*template_tran_index_value(i) + min_tran

Where:

i=2,…,number_of_indexes-1

The *template_tran_index_value* is given in a `define_template` command where the index values are treated as a ratio between 0 and 1.

This parameter must be used before the char_library command is run.

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_library -auto_index
```

scan_dummy_include_leakage_power

`<0 | 1>`	Request leakage power in the scan dummy. Default: `0`

Liberate can output leakage group data into the scan dummy cell. Set this parameter to **1** to enable this feature. When the parameter is set to `0` by default, it indicates not to output leakage power in the scan dummy.

This parameter must be used before the char_library command.

sdf_cond_equals

`<string>`	Specify the `sdf_cond` attribute style. Default: `""`

Use this parameter to specify the format for the *sdf_cond* construct to be used in output models such as Liberty, Verilog, and Vital models. This parameter supports the following values: `"=="`, `"== logical"`, `"== binary"`, `"==="`, `"=== logical"`, and `"===`

binary". This parameter is identical to and will replace the sdf_cond_equals parameter to write_library. It allows the sdf_cond_equals modification to be applied to all output modeling routines including write_library, write_verilog, and write_vital allowing all models to use the same modeling format for sdf_cond.

The write_library -sdf_cond_equals command will continue to be supported for backward compatibility, but if used, it will set this parameter such that any subsequent write_verilog or write_vital commands will also have this modification.

This parameter is sequence dependent. It only impacts the models written after setting this parameter.

This parameter must be used after the char_library command.

Example

```
# The following will result in an inconsistency between the .lib and .v model files.
write_library no_sdf_equals.lib
set_var sdf_cond_equals "=="
write_verilog sdf_equals.v
# The following will create consistent .lib and .v model files.
set_var sdf_cond_equals "=="
write_library sdf_equals.lib
write_verilog sdf_equals.v
```

sdf_cond_postfix

<string>	The postfix to use for complex conditional sdf_cond attributes on sequential cells. Default: ""

This parameter sets the postfix that Liberate uses for naming complex sdf_cond attributes on sequential cells (flip-flop or latch). If an sdf_cond attribute is complex (that is it has two or more operands) it will be replaced by a name of the format ${sdf_cond_prefix}#${sdf_cond_postfix} (see the parameter sdf_cond_prefix) where the # is a unique number for each sdf_cond within the cell. The sdf_cond attributes are used for writing conditional timing arcs in Liberty, Verilog and Vital formats (see write_library, write_verilog, and write_vital).

This parameter must be used before the char_library command.

Example

```
# Set the sdf_cond postfix
set_var sdf_cond_postfix "int_cond"
char_library
write_library my.lib
```

sdf_cond_prefix

`<string>`	The prefix to use for complex conditional `sdf_cond` attributes on sequential cells. Default: `adacond`

This parameter sets the prefix that Liberate uses for naming complex `sdf_cond` attributes on sequential cells (flip-flop or latch). If an `sdf_cond` attribute is complex (that is, it has two or more operands) it will be replaced by a name of the form `${sdf_cond_prefix}#${sdf_cond_postfix}` (see the sdf_cond_postfix parameter) where the **#** is a unique number for each `sdf_cond` within the cell. The `sdf_cond` attributes are used for writing conditional timing arcs in Liberty, Verilog and Vital formats (see write_library, write_verilog, and write_vital).

This parameter must be used before the char_library command.

Example

```
# Set the sdf_cond prefix
set_var sdf_cond_prefix "int_cond"
char_library
write_library my.lib
```

sdf_cond_style

`<0 | 1 | 2>`	Specifies the `sdf_cond` attribute. Default: `0` The `sdf_cond` attribute for complex conditional constraint arcs must be represented as a "parameter" that is the functional equivalent to the when condition.

0	Generates a unique parameter for each different complex condition (has merged states as in "state1 || state2") as follows: `${sdf_cond_prefix}#${sdf_cond_postfix}` See also the sdf_cond_prefix and sdf_cond_postfix parameters.
1	Generates an SDF variable name from the "when" condition for all complex conditions and for all conditional constraints such that the logic and operators will be replaced with `_AND_`, logic *or* with `_OR_`, logical *not* with `_NOT_`, `"("` with `OP_`, and `")"` with `CP_`. This constructed parameter name is prefixed with the value of the sdf_cond_prefix parameter and postfixed with the value of the sdf_cond_postfix parameter. For example, `"A * (B | !C)"` becomes `"adacond_A_AND_OP_B_OR_NOT_C_CP"`
2	Generates a constructed SDF variable name for all arcs using the naming convention as described in the setting of `1` above.

Note: `sdf_cond_style=1` applies to all conditional constraints while `sdf_cond_style=0` applies only to complex *when* conditions that have multiple operands.

This parameter must be used before the char_library command.

sdf_cond_style_for_constraints

`<0 | 1>` Controls whether the SDF conditions for constraints should follow the `-when` logic format when sdf_cond_style is set to `0`.
Default: `0`

- `0` All constraints are mapped to a variable name.

  Note: This value is provided for backward compatibility with the LIBERATE 18.1 ISR1 and prior releases.

- `1` During characterization, if the `sdf_cond_style` parameter is set `0`, the SDF conditions for constraints will follow the `-when` logic format. The string format that uses `ada_cond` will not be used.

This parameter must be used before the char_library command.

sdf_cond_variable_map

`{list of paired strings}`

Specifies a list of string pairs that map strings in the arc "when" to alternative strings in the arc `sdf_cond`.
Default: `{" " "" ( OP_ ) _CP ! _NOT_ && _AND_ || _OR_}`

This parameter must be used before the char_library command.

sdf_logic_and

`"string"` Sets the characters to use for denoting logic AND for `sdf_cond` attributes.
Default: `" & "` *(Notice the space on both sides of &)*

This parameter must be used before the char_library command.

sdf_logic_not

`"string"`	Sets the characters to use for denoting logic NOT for `sdf_cond` attributes. Default: `"~"`

This parameter must be used before the char_library command.

Examples

```
# Set the logic AND, OR and NOT to &&, || and !
set_var sdf_logic_and "&&"
set_var sdf_logic_or "||"
set_var sdf_logic_not "!"
```

sdf_logic_or

`"string"`	Sets the characters to use for denoting logic OR in `sdf_cond` attributes. Default: `" | "` *(Notice the space on both sides of |)*

This parameter must be set before the `char_library` command.

server_timeout

`<value>`	Specify unresponsive client message interval. When the server is waiting for a response from a client, it prints out a message when it has waited for the timeout specified by this parameter. The message repeats every timeout duration. Default: `10800` (seconds)

This parameter must be used before the `char_library` command.

Sample Message:

The simulation on client `<client>` has been running for 180 minutes. The simulation might be stalled, waiting for a license, or terminated. If the simulation has terminated and the client is not responding, you might need to restart the characterization job. Check the simulation status, for example, check the `sim.lis` file if using external simulator.

set_logic_condition_resolve_conflicts

`<0 | 1>` Specifies how to handle conflicts between the set_logic_condition command and the define_arc `-when` command. The `set_logic_condition` command can be used to set logic conditions.
Default: `0`

- `0` If the `-when` logic expression of a `define_arc` command conflicts with the logic expression specified by a `set_logic_condition` command, Liberate will generate an error and skip the corresponding arc.
- `1` When Liberate detects a logic conflict between the `define_arc -when` command and the `set_logic_condition` command, it will apply the logic expression of the `define_arc -when` command, and then apply the logic expression of `set_logic_condition` command after removing any conflicting pin assignment from it. This prevents the arc from being skipped due to the logic conflict between the two commands.

This parameter must be used before the char_library command.

Example

```
set_var set_logic_condition_resolve_conflicts 1
```

set_var_failure_action

`<warning | error>` Notifies the `set_var` command how to consider a failure.
Default: `warning`

- `error` — When a `set_var` fails, an error message is issued and subsequent commands which would result in characterization or library generation (for example, char_library) are suppressed. Subsequent `set_var` commands are still allowed so they can be checked for correctness.
- `warning` — A warning is issued when `set_var` fails. The failed `set_var` is ignored and execution continues.

This parameter must be set before any other set_var command.

sim_default_engine

`<string>` Specifies the default simulation engine to use when characterizing libraries.
Default: "SKI"

If you want to use the Alspice simulation engine, you can set it to "alspice". The default simulation engine specified by this parameter is overridden by the one specified using the `char_library -extsim` command option.

This parameter must be set before the char_library command.

sim_duration

`<value>` Sets the maximum allowed simulation time to be used during transistor-level (SPICE) simulation. This parameter puts a limit on the total simulation duration. It acts as a global override.
Default: For `-io` mode (see char_library `-io`): `1e-5`;
all other modes: `1e-7`

This parameter must be used before the char_library command.

Example

```
# Set the parameter 'sim_duration' to 1e-5 seconds
set_var sim_duration 1e-5
```

sim_estimate_duration

<0 | 1>	The maximum simulation duration. Default: 1 (Estimate the simulation duration)

Use this parameter to disable the algorithm in Liberate that will estimate the expected simulation duration. Set this parameter to **0** to disable the estimation. Once disabled, the simulation will run for the time specified by the control parameter sim_duration.

This parameter must be used before the char_library command.

Example

```
# Set the parameter 'sim_duration' to 1e-5 seconds
set_var sim_estimate_duration 0
```

sim_init_condition

<hybrid | ic | ic_except_dc | ic_except_dc_plus_leakage>

Specify initial condition method when presenting spice decks to the simulator.
Default: hybrid

The **ic** option requests Liberate to always use the SPICE .ic command in generated spice decks. This option is not compatible with Leakage and CCS DC sweep data characterization. The ic_except_dc option requests Liberate to use .ic for all generated spice decks except those which use a DC analysis. The ic_except_dc_plus_leakage option requests Liberate to use .ic to perform leakage characterization. The hybrid option is the default. When set to hybrid, Liberate will use .ic for any floating nodes and .nodeset everywhere else.

This parameter must be used before the char_library command.

Example

```
set_var sim_init_condition ic_except_dc
```

sim_init_condition_estimation_mode

`<0 | 1>`	Specify initial condition method. Default: `1`

Use this parameter to select the algorithm that Liberate will use to initialize internal nets.

When set to **0**, Liberate may set incorrect signal swing rails for internal nets when a cell has multiple supplies.

When set to **1**, a modified algorithm seeks to remedy this issue by propagating supplies twice. In the first pass, vdd/gnd will only propagate through pmos and nmos channels. In the second pass, Liberate will propagate vdd/gnd through all channels for those nets that didn't get set in the first pass. To restore the behavior of the 2.3p2 and prior releases, set this parameter to `0`.

This parameter must be used before the char_library command.

sim_init_duration

`<value>`	The initialization simulation duration. Default: `500e-12 (500ps)`

Liberate can provide initial conditions, as needed for a vector to function properly to SPICE. The SPICE engine will simulate for the specified duration and report the internal node voltages to Liberate to use during subsequent SPICE simulations of the vector.

Note: This parameter will have an effect only when used in combination with `sim_use_init_duration`.

When the `ramp_vsrc` parameter is set, the `sim_init_duration` parameter will determine the amount of time Liberate will simulate to determine the internal node voltages. If ramp_vsrc is set, then this parameter must also be set. The `leakage_sim_duration` parameter will override this parameter for all leakage simulations.

This parameter must be used before the char_library command.

Example

```
# Set the parameter 'sim_init_duration' to 1e-9 seconds
set_var sim_init_duration 1e-9
```

sim_power_duration_extend

`<value>`	Specify an increment to add to the power simulation. Default: `0` (Specified in seconds)

Use this parameter to increase the spice dynamic power simulation duration by a user defined increment in seconds. This extension is only applied when combined with `power_subtract_leakage` set to `2`. The power simulation duration will be capped at the value of the sim_duration.

This parameter must be used before the char_library command.

Example

```
# Increase the power simulation duration by 10pS
set_var sim_power_duration_extend 100e-12
```

sim_use_init_duration

`<0 | 1 | 2>`	Turn on the sim_init_duration algorithm. Default: `0`

Liberate can provide initial conditions to spice, as needed, for a vector to function properly. The spice engine can be used to validate the initial conditions using a transient simulation. The spice engine will simulate for the **sim_init_duration** and report the internal node voltages to Liberate which will use them during subsequent spice simulations of the vector. Set this parameter to **1** to use this algorithm. This can help to work around issues where the spice engine is having problems finding a dc solution. When set to **2**, the pre-simulation for capturing internal conditions for all internal nodes will have its supply ramped from 0 to vdd, similar to **leakage_ramp_vsrc** 1 – except this is done for timing simulations. Default: **0** (do not use spice transient solution)

This parameter must be used before the char_library command.

Example

```
# Set the parameter 'sim_use_init_duration'
set_var sim_use_init_duration 1
```

simultaneous_switch

`<0 | 1>`	Enables simultaneous input switching analysis. Default: `0`

Use this parameter to enable analysis for simultaneous switching inputs. Any cell that has any of the following will **not** be analyzed for simultaneous switching inputs: *ff* group, *latch* group, *statetable* group, *clock* attribute, *three_state* attribute, direction attribute of *inout*, *rise_constraint* group, and *fall_constraint* group.

Set to one ("1") to enable simultaneous input switching for combinational gates. When enabled, Liberate automatically finds one or more worst or best case simultaneous input switching vectors for each arc. Large fan-in cells will have a substantial performance penalty. It is recommended to set the parameter `conditional_arc` to 0 to substantially reduce simultaneous switching characterization time.

Use the `set_default_group` command to set the default timing criteria to a minimum or maximum during (simultaneous switching) characterization to control the reported timing values.

This parameter must be used before the char_library command.

Example

```
# Enable simultaneous switching analysis.
set_var simultaneous_switch 1
```

simultaneous_switch_offset

`<value>`	Offset to be used between the related_pin and the simultaneously switching side inputs. Default: `0` (seconds)

Set this parameter to specify a timing offset to be applied between related_pin and the simultaneous switching side inputs. The offset must be greater than or equal to zero so that the side inputs will switch with or after related_pin. All switching side inputs will use the same transition time.

This parameter will be honored at all times, even when the parameter **simultaneous_switch** is not enabled. This parameter has a default value of **0** so Liberate will not offset MIS (Multiple Input Switching) signals automatically.

This parameter must be used before the char_library command.

Example

```
# Set the side inputs to switch with an offset of 5pS.
set_var simultaneous_switch_offset 5e-12
```

simultaneous_switch_use_arc_when

<0 | 1 | 2>	Defines how the -when argument of the define_arc command is applied to simultaneous input switching detection. Default: 0 (ignore define_arc *when*)

- If simultaneous_switch_from_cell_when is 0, this parameter has no effect.
- If simultaneous_switch_from_cell_when is enabled (1 or 2), this parameter works as follows:

  0: (Default). In this case, the following two conditions apply:

  If simultaneous_switch_from_cell_when is 1, then no pin listed in the define_arc -when is permitted to switch.

  If simultaneous_switch_from_cell_when is 2, then the define_arc -when is completely ignored if the side pin is switching.

  1: The define_arc -when will be observed before the side pins switch.
  2: The define_arc -when will be observed after the side pins switch.

Note: The recommended value of this parameter is 0.

This parameter must be used before the char_library command.

Example

```
# Set the parameter 'sim_use_init_duration'
set_var simultaneous_switch_use_arc_when 1
```

simultaneous_switch_worst_vector

`<1 | 4>` Number of slew/load simulations used in determination of worst simultaneous switch vectors.
Default: `1`

- `1` Use mid-point of slew/load indexes for binning.
- `4` Restores the behavior of the 2.3 and earlier releases where the following four slew/load simulations were used:
  - `min_slew/min_load`
  - `min_slew/max_load`
  - `max_slew/min_load`
  - `max_slew/max_load`

This parameter must be used before the char_library command.

Example

```
# Use 4 slew/load simulations for binning worst simultaneous
# switching vectors
set_var simultaneous_switch_worst_vector 4
```

ski_alter_mode

`<1 | 2 | 3 | 4>` Controls the method for generating SPICE decks used in constraint acquisition.
Default: `3`

`1`	Use external simulator compatible settings (recommended for matching Spectre and so on)
`2`	Use Alspice-compatible settings (recommended for matching Alspice)
`3`	Use external simulator compatible settings (recommended for matching Spectre and so on) similar to the setting of 1 above, but with optimizations to improve run time (default).
`4`	Use Alspice-compatible settings (recommended for matching Alspice) similar to the setting of 2 above, but with optimizations to improve run time.

Liberate can employ several different methods for generating SPICE decks during the bisection search for constraint acquisition. Due to numerical differences within the meta-stable region, these can result in slightly different constraint results.

This parameter has an effect only if `ski_enable=1`. See External Simulator Options and Settings for recommended settings.

Note: The `ski_alter_mode` parameter setting might affect performance. However, `ski_alter_mode=2` should be faster.

This parameter must be used before the char_library command.

ski_clean_mode

`<0 | 1 | 2>` Enables clean up of the shared memory environment used by SKI.
Default: `0` (not enabled).

- `0` Perform no clean up.
- `1` Perform standard cleanup by adding the `-a` option to the clean-up command. (Recommended)
- `2` Same as 1, but log additional debugging information by adding the `-d` option to the clean-up command. This is useful to debug situations where SKI is consistently being disabled on certain farm machines.

Enabling `ski_clean_mode` directs Liberate to clean up semaphore resources that are owned by you but are not in active use. This feature can augment or replace similar capabilities enabled in `altos_init` files. The clean-up is performed by running the script in `$ALTOSHOME/bin/clean_sm.sh`, which may be independently modified or updated, if required.

When `ski_enable=1`, Liberate and Spectre communicate through the Spectre Kernel Interface (SKI) using shared memory and semaphores. Because these resources are both system-wide and limited, if insufficient resources are available to run SKI, then Liberate disables SKI and uses the standard Spectre interface. While this yields accurate results, there is a performance penalty.

Also, considering that these resources follow the standard rules of Unix permissions, you cannot clean up resources belonging to another user. The resources must be cleaned up by the owner or by root.

This parameter must be used before the char_library command.

ski_compatibility_mode

`<0 | 1>` Enables consistency between standalone Spectre and Spectre-SKI.
Default: `0`

`0` Each individual SKI control parameter is set manually.

`1` Set the following parameters to corresponding values:

- `ski_power_subtract_output_load_match_extsim 1`
- `alspice_power_subtract_output_load_match_extsim 1`
- `ski_alter_mode 3`
- `ski_mdlthreshold_exact 1`
- `capacitance_save_mode 0`

The dependent parameters listed above are under Liberate control and cannot be overridden while `ski_compatibility_mode` is set to 1.

This parameter must be used before the char_library command.

ski_enable

`<0 | 1>`

Enables the Spectre Kernel Interface, a tight integration between Liberate and Spectre that provides a substantial performance improvement. See External Simulator Options and Settings for recommended settings. SKI can be enabled or disabled for a specific arc (see set_var) as long as it is enabled globally. If SKI is globally disabled, it cannot be enabled locally using `set_var`. In addition, then the setting overrides any arc-specific settings with a warning.
Default: `0` (off)

- `0` — Disables the Spectre high-performance (SKI-based) interface.
- `1` — Enables the Spectre high-performance (SKI-based) interface.

Important Points to Note

- SKI support was not available before Spectre version MMSIM10.1 ISR20, MMSIM11.1 ISR8, or equivalent newer versions. See the `${ALTOSHOME}/README` file for the qualified version of Spectre. As SKI uses inter-process communications, it is strongly recommended to use the Spectre SKI version that was qualified for the LIBERATE release.
- To use Spectre as the circuit simulator, the `char_library -extsim spectre` option must be set.
- SKI is supported only in the ***parallel packet*** flow.

This parameter must be used before the char_library command.

Example

```
set_var ski_enable 1
set_var -cell DFF -type mpw -pin RN ski_enable 0
```

ski_mdlthreshold_exact

`<0 | 1>`

Enables the Spectre option `mdlthresholds=exact` when `ski_enable=1`.
Default: `0` (not enabled).

SPICE waveforms normally use given relative and absolute tolerances to determine the time-steps in a waveform. The Spectre option, `mdlthresholds=exact`, has the ability to place a time-point exactly on a threshold crossing (for example, delay and slew thresholds) at the expense of some performance. This option is set to `exact` by default in Spectre, but is set by default to `interpolated` in SKI.

`0` SKI uses a setting similar to the `mdlthresholds=interpolated` behavior of Spectre.

`1` SKI uses a setting similar to the `mdlthresholds=exact` behavior of Spectre Circuit Simulator Measurement Description Language (MDL). (Recommended for consistency with standalone Spectre default behavior).

For more details and additional settings, see the Correlation between stand-alone Spectre and SKI section.

This parameter must be used before the char_library command.

ski_meas_mode

`<0 | 1>` Specifies whether to use the Spectre Kernel Interface (SKI) flow or the LXI flow.
Default: `0`

- `0` Use the SKI flow to get waveforms using Spectre and calculate results using Liberate.
- `1` Use the LXI flow, add `.meas` statement, and get results using those measures.

  For more information, see the LXI Support section.

This parameter must be used before the char_library command.

ski_power_subtract_output_load_match_extsim

`<0 | 1>` Lets you switch between charge-based or voltage-based power calculation.
Default: `0` (use voltage-based power for subtraction)

- `0` SKI uses charge-based power calculation. (Default)
- `1` SKI uses voltage-based power calculation. (Recommended)

Liberate can calculate power based on charge or voltage. External simulations always use voltage-based power calculations. A well-designed circuit should yield similar results when switching between the two methods.

To remove power correlation differences on sensitive circuits based on power calculation methodology, it is recommended to use the same approach for both SKI and external simulations.

This parameter must be used before the char_library command.

ski_reset_cnt

`<integer>`	Force reset of Spectre-SKI memory usage. Default: `50000` (Reset after 50,000 iterations)

When SKI is enabled, Liberate will start a separate Spectre simulation process. The memory footprint of this process can change over time (both increasing and decreasing). If the process grows too large (more than available memory), then performance can degrade. This parameter will force a memory reset after the specified simulation iteration count. This parameter should be only used as a safety measure in the case when memory usage grows extremely fast or for memory debugging. Small values will increase runtime. Values below 1000 are not recommended.

This parameter has an effect only if `ski_enable=1` (Spectre-SKI is enabled). See External Simulator Options and Settings.

This parameter must be used before the char_library command.

ski_sync_method

`<0 | 2>`	Controls the method for SKI data-syncing. Default and recommended: `2`	
	`0`	Use Semaphore Arrays for data-syncing.
	`2`	Do not use Semaphore Arrays for data-syncing.

This parameter must be used before the char_library command.

ski_use_large_memory

`<value>`

Set this parameter to a higher value if you get a message similar to the following.
Number of simulation time points at t=8.48005e-08 (s) exceeds size of pre-allocated waveform data storage.

Note: This parameter is used when SKI flow is enabled. (see ski_enable).

The allowed values for this parameter are `0` to `10`.
Default: `3`

These parameters must be used before the `char_library` 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; liberate
```

`1` Submit runs to the batch tool using:

```
liberate
```

This parameter may be needed when using a batch system that cannot accept a batch command containing multiple commands separated by a semicolon

This parameter must be used before the char_library command.

slew_lower_fall

`<value>`

The % point on the cell output waveform to output falling output transition times to.
Default: `0.2 (20%)`

This parameter must be used before the `char_library` command.

slew_lower_rise

<value>	The % point on the cell output waveform to output rising output transition times from. Default: 0.2 (20%)

This parameter must be used before the char_library command.

slew_normalize

<0 | 1>	Report the normalized or measured slew. Default: 1 (Report normalized slew)

By default, Liberate will measure the output slews using the measure_slew* parameter values and then normalize this measured value to the slew* reporting values. To have Liberate report the actual measured value, set this parameter to 0.

This parameter must be used before the char_library command.

slew_upper_fall

<value>	The % point on the cell output waveform to output falling output transition times from. Default: 0.8 (80%)

slew_upper_rise

<value>	The % point on the cell output waveform to output rising output transition times to. Default: 0.8 (80%)

The above parameters are used to control how output transition times are stored in the delay tables. These parameters can be set differently from the equivalent measurement thresholds, in which case the measurements are normalized to fit the appropriate output transition slew thresholds.

If these parameters are set symmetrically, the output library that is generated will include the slew_derate_from_library attribute and the slew_*_threshold_pct_* attributes will be set to the equivalent measure_slew_* parameters (x 100). If they are not

symmetrical then the `slew_derate_from_library` attribute will be omitted and the `slew_*_threshold_pct_*` attributes will be set to the equivalent `slew_*` parameters (x 100).

This parameter must be used before the `char_library` command.

Example

```
# Set the output transition thresholds to 10-90%
set_var slew_lower_fall 0.1
set_var slew_upper_fall 0.9
set_var slew_lower_rise 0.1
set_var slew_upper_rise 0.9
```

sort_cells

`<value>`	Set this to a 0 to disable sorting cells alphabetically. Default: `1` (Sort the cells.)

This parameter controls the sorting of cells into the *output.lib* file. The default is to sort all cells alphabetically.

The pins within a cell are sorted with outputs first, then bidirectional pins followed by input pins. The pins are sorted alphanumerically within each of those groups.

This parameter may be used after **char_library**.

Example

```
# Do not sort cells
set_var sort_cells 0
```

sort_groups_under_pin

`<value>`	Set this to a 0 to disable sorting pin timing groups by "when" alphabetically. Default: `1` (Sort alphabetically.)

Use this parameter to request Liberate to sort the timing groups alphabetically using the "when" state description. By default, the timing groups under a pin are sorted alphabetically . This command is useful when an arbitary order is desired for the timing groups under a pin.

This parameter can be used after **char_library**.

Example

```
# Sort the timing groups
set_var sort_groups_under_pin 1
```

sort_pins

<0 | 1 | 2>	Specifies the sorting of pins for each cell when writing the library. Default: 1 (Sort)
0	Do not sort the pins.
1	Sort the pins for each cell in alphabetical order. (Default)
2	Sort the pins for each cell in reverse alphabetical order.

This parameter can be used together with pin_type_order to specify that pins be sorted within groups, that is, keep all input pins together and sort them; keep all output pins together and sort them; and so on.

This parameter must be set before the write_library command is run.

sort_pins_under_when

<0 | 1>	Enable sorting of pins in a WHEN. Default: 1 (Sort pins alphabetically)

Liberate can sort the pins in a *when* condition in the output library. When set to 1, Liberate sorts pin names within the *when* string in the sequence given below:

```
input pins - alphabetical
bidi pins - alphabetical
output pins - alphabetical
```

This parameter must be set before the char_library command is run.

Example

```
# Disable sorting the pins
set_var sort_pins_under_when 0
```

spectre_dash_log

`<0 | 1>` — Enables the Spectre `-log` command line option when set to 1. Default: `1`

Set this parameter to 0 to disable the automatic addition of `-log` to the `extsim_cmd` option when the `-extsim` Spectre option of the char_library command is used.

When using Spectre, Liberate automatically adds `-log` to the `extsim_cmd`_option. This is done to minimize disk usage because the stdout/stderr is saved by Liberate into a file called `sim.lis` and an additional log file is not needed. This parameter controls whether the Spectre `-log` option will be added when Spectre is used. It may be desirable during debug to view the Spectre log file because it contains additional messages that do not print to stdout/stderr. Set this parameter to 0 to save both the Liberate `sim.lis` and the Spectre `sim.log` files.

This parameter must be set before the char_library command is run.

spectre_use_char_opt_license

`<0 | 1>` — Controls use of `Spectre_char_opt` license feature. Default: `1` (use the licensed feature if it exists)

This parameter enables Liberate to checkout `Spectre_char_opt` licenses upfront instead of MMSIM licenses, which are only checked out by Spectre. This will minimize the overhead associated with checking out of Spectre licenses.

Important

The `Spectre_char_opt` license feature is restricted for Liberate characterization flows and cannot be used for other Spectre simulations. This license feature must be purchased and your license file must contain the `Spectre_char_opt` feature.

`0`	Do not use the `Spectre_char_opt` feature. Use this setting to disable the use of the `Spectre_char_opt` license feature. You can also use this setting if your license file does not contain the `Spectre_char_opt` feature. In this case, Liberate will automatically reset `spectre_use_char_opt_license` to `0`. This will prevent a slowdown in performance due to Liberate checking for a nonexistent license feature.
`1`	Use the `Spectre_char_opt` feature if it exists. To restrict license checkout to the `Spectre_char_opt` license feature, use the packet_require_spectre_char_opt parameter.

This parameter must be set before the char_library command is run.

spectre_use_mmsim_token_license

`<0 | 1>` Enables Liberate to checkout the `Virtuoso_Multi_mode_Simulation` licenses (also known as MMSIM tokens) upfront to minimize the overhead associated with checking out of Spectre licenses for each simulation. Default: `1`

`0`	Do not checkout the `Virtuoso_Multi_mode_Simulation` license feature.

`1`
: Liberate can checkout the `Virtuoso_Multi_mode_Simulation` license feature. The quantity of MMSIM tokens that need to be checked out for each client depends on the Spectre command-line options specified with the `extsim_cmd_option` parameter according to the table given below.

Spectre Command-Line Options	Required MMSIM Token Quantity for Each Client
No options specified	1
`+aps -mt`	2
`+aps`	6 (APS default behavior enables multi-threading)
`+aps +mt=2 ... 4`	4
`+aps +mt=5 ... 16`	6
`+aps +mt=17 ... 32`	8

Important

If the `Spectre_char_opt` license feature is found on the license server, Liberate resets `spectre_use_mmsim_token_license` to `0` giving preference to the `Spectre_char_opt` license feature.

Other parameters that affect licensing are packet_require_spectre_char_opt and spectre_use_char_opt_license.

This parameter must be set before the char_library command is run.

spice_character_map / spectre_character_map

`"list of character pairs"`
: List of characters to be mapped when building SPICE decks. Default: `""` (No character mapping)

Use this parameter to tell Liberate to map characters in the netlist to alternate characters in the SPICE decks. The string is comprised of character pairs, where the first character is replaced with the second character. Multiple mappings are supported.

Important

Spectre-SKI does not support this feature.

The list must *not* contain whitespace.

The list *must* contain an even number of "paired" characters.

This is independent of the external simulator used. The mapping is applied during `read_spice`. If "`read_spice -format spectre`" is used, then the `spectre_character_map` is applied. Otherwise, the `spice_character_map` is applied.

We highly recommended modifying the extraction tool to *not* use undesirable characters as part of any net name. It is possible that the resulting mapped deck could have name collisions. This can be especially true if using SPICE-formatted netlists with characters used in bitwise operations (for example, &, |) with Spectre.

This parameter must be set before the read_spice command is run.

Example

```
# map the character '/' to '_' ... then map '&' to 'a'.
set_var spice_character_map "/_&a"
```

spice_delimiter

`"string"`	Character(s) used to indicate hierarchy in the input SPICE netlists. Default: `"."`

This parameter specifies the hierarchy delimiter in the SPICE format netlists loaded into **read_spice**. It can be a single or multiple character string. Every character in this parameter 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.

This parameter must be set before the char_library command is run.

Example

```
# Set the SPICE delimiter to |
set_var spice_delimiter "|"
```

spice_delimiter_replacement

`"string"`	Character used to replace hierarchy in the input SPICE netlists. Default: `"_"` (Underscore character.)

By default the parameter `extsim_use_node_name` is `0`, and the node id's in the spice decks for the external simulator are represented in numerical form. If `extsim_use_node_name` is set to `1`, some characters (for example, : / .) could be interpreted has a *hierarchical delimiter* even if the deck itself has been flattened. In some cases there are characters that cannot be allowed by the SPICE netlist interpreter (for example, as voltage source name or node). Use this parameter to specify the character that will be used to replace those delimiter characters. Default "_".

This parameter must be set before the char_library command is run.

Example

```
# Set the SPICE delimiter replacement character to #
set_var spice_delimiter_replacement "#"
```

spice_instance_name_require_x_prefix

`<0 | 1>`	Specifies that SPICE instance names must begin with the character "X". Default: `1` (on)

Instance names in SPICE netlists are typically required to begin with the character "X". However, Spectre does not enforce this, so turn this parameter off to allow netlists with instance names not prefixed by "X".

This parameter must be set before the read_spice command is run.

Example

```
set_var spice_instance_name_require_x_prefix 0
```

spice_logical_netname_mode

`<0 | 1>`

Use logical wire name instead of shortest node name.
Default: `0`

Note: The default value is overwritten to 1 in the `write_vdb` flow.

When `spice_logical_netname_mode` is set to 1, instead of setting the logical net name to the shortest node name, Liberate will set the logical wire name using the name of the attached instance.

Setting this parameter to `1` when the vector database (VDB) flow is used with switch cells and multiple PVTs with different extracted netlist are characterized resolves compilation errors such as having inconsistent names in different PVTs when use of the same logical name is desired across PVTs.

This parameter must be set before the char_library command is run.

split_pvt_prechar_job

`<0 | 1>`

Controls creation of a precharacterization job to run auto-index for each cell-PVT combination after the counting job and before the simulation job.
Default: `0`
Recommended: `1`

Note: This parameter works when Bolt job distribution system is used in multi-PVT mode.

`0`	Disable the feature.
`1`	Enables creation of a precharacterization job. This feature helps to avoid rerunning auto-index in all simulation jobs and thus, improve performance.

This parameter must be set before the char_library command is run.

subtract_hidden_power

`<0 | 1 | 2>` Control the subtraction of hidden power from internal power.
Default: `0` (Do not subtract hidden power.)

- `0` Do not subtract hidden power.
- `1` Subtract hidden power in sequential cells.
- `2` Subtract hidden power for all cells.

This parameter must be set before the char_library command is run.

subtract_hidden_power_consider_all_supplies

`<0 | 1>` Controls whether to subtract all of the hidden power groups that overlap with the switching power if the cell has multiple supplies such as in a level shifter. This parameter can be used only when the `subtract_hidden_power_use_default` parameter is set to `2`.
Default and recommended: `1`

- `0` Subtracts only the last hidden power group among the hidden power groups that overlap (usually have no when condition). This can result in wrong results if there are multiple supplies in the cell because every supply corresponds to one such hidden power group. This setting is provided for backward compatibility to the LIBERATE 15.1 ISR5 release.
- `1` Subtracts all hidden power groups corresponding to all supplies that overlap with the switching power.

This parameter must be set before the char_library command is run.

subtract_hidden_power_scalar_mode

`<0 | 1>` Determines whether the scalar power groups should be replaced with a copy of the closest matching hidden power data.
Default: `0`

- `0` Scalar zero groups are not replaced.
- `1` Scalar zero power groups of the active input edge are replaced with a copy of the closest matching hidden power data. For this to happen, subtract_hidden_power should be >`0`.

  Also, if subtract_hidden_power_use_default is set to `1`, the values will be from the default group, if default groups are enabled. If `subtract_hidden_power_use_default` is not set to `1`, the values may be from a closely matching `when` condition used in subtraction.

When a cell needs hidden power subtraction due to overlapping power conditions (for example, MBFF), you must set `subtract_hidden_power` to a value greater than `0`. When subtract hidden power is enabled, matching is employed to choose the closest hidden power group to the switching power group. The matching algorithm is controlled by `subtract_hidden_power_use_default`.

Normally a switching power condition is not modeled under an input pin group, but only through the output pin power groups. Depending on cell functionality or the user's cell template, the input pin group may have placeholders of ***scalar*** power groups with a value of zero for syntactical correctness when switching power occurs and hidden power is not characterized. After hidden power subtraction is performed, if there are any scalar power groups under the input pin that is considered an active edge, these will now need to be populated with the hidden power data used in subtraction to account for the portion of the shared switching power in power simulation. The `subtract_hidden_power_scalar_mode` parameter setting provides the functionality to replace the scalar power groups.

This parameter must be set before the write_library command is run.

subtract_hidden_power_use_default

<0 | 1 | 2 | 3> Controls what value of hidden power should be used when subtracting hidden power from switching internal_power. Default: 0

0 Liberate chooses one of the non-conflicting state-dependent hidden power groups to subtract. If all hidden states conflict with the switching state, hidden power is not subtracted. Following are examples of:

Conflict	Non-Conflict
CK->Q when "!SE" with CK hidden when "SE&D".	CK->Q when "!SE" with CK hidden when "!SE&D"
	CK->Q when "" with CK hidden when "!SE&D"

1 The default hidden power value is used for subtraction from the switching power. If no default hidden power is found, a warning is issued and the related subtraction is skipped. When using this setting, it is recommended that the force_default_group parameter is also set to a 1.

2 (Recommended) Liberate uses the following priority for subtraction of the default hidden power:

- Subtract the hidden power matching the simulation vector, if it exists
- Subtract the default hidden power, if it exists

If no default hidden power is found, a warning is issued and the related subtraction is skipped. When using this setting, the WHEN state must also have exact or partial overlap with the switching power state because the power tool uses this while adding the default hidden power.

3	Extends mode 2 to match switching and hidden WHEN conditions if the hidden output pin conditions are compatible with the switching direction.

Note: This parameter will have no effect if subtract_hidden_power is set to 0.

This parameter must be set before the write_library command is run.

supply_define_mode

<0 | 1 | 2>

Changes the way supplies are recognized. When set, there are no default supplies. Therefore, all supplies must be identified using the set_vdd and set_gnd commands.
Default: 0

0	Determines the supplies by tracing the connectivity. All supplies must connect to a transistor. **Note:** The default supplies are: VDD, VCC, and VSS.
1	Determines the supplies from module wires instead of tracing connectivity to transistors. This setting is useful in matching legacy libraries and when there are unconnected supplies that must be modeled in the library. **Note:** There are no default supplies. All supplies must be identified using set_vdd and set_gnd.
2	Adds a missing power pg_pin construct to the library for the cell, if needed. If a cell has pins, it is required that the cell should have at least one power and one ground rail. If the cell has no pins, Liberate does not add any missing pg_pin constructs. The setting of 2 is an extension to the setting of 1 described above.

This parameter must be set before the set_operating_condition command is run.

supply_info

`<0 | 1>`

Controls the printing of messages that summarize the values set for the following commands: `set_vdd`, `set_gnd`, `set_pin_vdd`, and `set_pin_gnd`.
Default: `0`

Use this parameter if the correctness of pin voltage settings needs to be checked.

`0`	Does not print the messages.
`1`	Prints the messages.

Example

```
set_var supply_info 1
```

Liberate will print messages such as following:

```
(set_vdd): Pin 'vdd' is set to 20 V for cell 'test1' -no_model 'False' -ignore_power
'False' -type 'internal' -attributes {} -combine_rail 'False' -include {}
(set_pin_vdd): Pin 'A1' is set to 3 V for cell 'level_shifter_3to1' -add_supply
'True' -leakage_add_to_supply '' -supply_name {VDD3}
```

switch_cell_bounded_dc_current

`<0 | 1>`

Enables bounding the DC current sweep between supplies.
Default: `1`

Set this parameter to limit the DC current sweep for power switch cells.

`0`	Ensures that the DC current sweep reported for the power switch cells will use the same voltage range as the CCSN DC current.
`1`	Ensures that the DC current sweep will be limited to the gnd and vdd rail voltages.

This parameter must be set before the char_library command is run.

switch_cell_dc_current

`<0 | 1 | 2>` Includes or omits `dc_current` from switch cells.
Default: `1` (Include dc_current.)

- `0` Omits the `dc_current` from switch cells in an NLDM library. If CCSN is requested, the `dc_current` is not omitted. This can be useful to prevent characterization of the `dc_current` for switch cells when using a separate Liberate run to characterize leakage.
- `1` Includes `dc_current` in the switch cells when the `-internal_supply` option of the `define_cell` command is provided.
- `2` Always omit the `dc_current` from switch cells. Neither NLDM nor CCSN style libraries will include `dc_current`.

This parameter must be set before the char_library command is run.

switch_cell_dc_current_output_offset

`<value>` Specify voltage offset value used in switch cell dc_current.
Default: `0`

Use this parameter to specify an absolute voltage value that will be used to offset the output swing range of the *dc_current* constructs of switch cells. The default offset value is **0**. The formula applied when sweeping the *dc_current* is described below.

- input sweep range: gnd(input), vdd(input)
- output sweep range: gnd(output)+offset, vdd(output)-offset

This parameter must be set before the char_library command is run.

switch_cell_internal_net_name

`""` | `"use_dspf_star_net"` | `"`*`User_Name`*`"`

Chooses between a created name and a real net name for the `related_switch_pin` attribute.
Default: `""`

`""` Specifies to use a Liberate-generated dummy name like `"XMM<>_altosg<>"`.

`"use_dspf_star_net"` Defines to use the logical net name specified by `*|NET` in a DSPF format netlist file. The `*|NET` name should also correspond to a physical net in the RC network.

`"`*`User_Name`*`"` Provides a user-specified net name. This must be specified in a `set_var` command with the `-cell` and `-pin` options where the pin is the switched-mode power supply pin name.

Note: When you use this parameter, ensure that the extsim_use_node_name parameter is also set.

This parameter must be set before the char_library command is run.

Example

```
set_var switch_cell_internal_net_name "use_dspf_star_net"
set_var extsim_use_node_name 1
```

switch_cell_internal_node_timing_arc

`<0 | 1>` Generates an input to internal switch node timing arc.
Default: `0`

`0`	Skips generation of the timing arc of the internal supply pin. The timing arc of the internal switch node is copied from the timing arc of the output pin.
`1`	For power switch cells, generates input to internal switch node timing arc. When this parameter is set to `1`, the conditional `ccsn_*_stage` is not outputted.

This parameter must be set before the char_library command is run.

switch_cell_powerdown_function

`<0 | 1>` Specifies whether to generate the `power_down_function` on switch cells.
Default: `1`

`0`	Do not generate `power_down_function` for any cell.
`1`	Generate the `power_down_function`.

Note: Although this parameter has `switch_cell` in its name, it applies to all cells.

This parameter must be set before the char_library command is run.

switch_function_attr_mode

`<-1 | 0 | 1 | 2>` Selects the algorithm for determining the switch cell function. Default and recommended: `0`

`-1`	Form `switch_function` by primary input and unateness in the `ccsn_last_stage` group(s) of the internal supply pin in ldb
`0`	Use the three_state function of the internal supply pin in ldb as the `switch_function`.
`1`	Form `switch_function` by generating logic function with a CCC function identification algorithm and adding logic 'not' to invert the function;
`2`	Generates `switch_function` by using BDD to invert and simplify the identified function for the internal supply pin.

Notes:

- Setting of `-1` can be incorrect when stacked MOS are between real power and virtual power.
- Setting of `0` can be effective both in the read_ldb and write_library flow and in the char_library and write_library flow. This mode is the default mode.
- Setting of `1`, `2` can be effective only in the `char_library` then the `write_library` flow, but not in the `read_ldb` then the `write_library` flow. This is because the function identification algorithm requires circuit module.

This parameter must be set before the char_library command is run.

template_unique_power_mode

`<0 | 1>` Default: `0`

This parameter should never be set manually. It is set automatically in a template file that is created when the `-unique_power` ***argument of the*** `write_template` ***command is used. When the*** `-unique_power` ***argument is used, depending on the reference library, additional power arcs might be generated. In some libraries, this can result in the following:***

- Many more power arcs are generated because the modeled states ("when" conditions) differ between delay and power.
- Some power arcs are invalid because the reference library contains data duplication between rise and fall power that did not come from a valid simulation.

When this parameter is set to 1, the above issues are addressed, resulting in correct power values and run times as fast as without using the `-unique_power` argument.

This parameter must be set before the char_library command is run.

test_cell_at_end

`<0 | 1>` Control the position of the `test_cell` group.
Default: `1` (move the `test_cell` group)

Set this parameter to `0` to not move the `test_cell` group to the end of each cell in the library output by write_library rather than the current position which is after the leakage data and before the pin data.

This parameter must be set before the `write_library` command is run.

test_cell_filter_attributes

`<0 | 1>` Specifies the attributes that need to be removed from the `test_cell` group. This parameter can be used to specify multiple cells, pins, attributes by using the `-cell`, `-pin`, and a string of attributes.
Default: `""`

This parameter must be set before the write_library command is run.

Example

```
set_var test_cell_filter_attributes "attrA"
```

Removes attribute `"attrA"` from the `test_cell` group for all pins in all cells.

```
set_var -pin "A" test_cell_filter_attributes "attrA attrB"
```

Removes attributes `"attrA"` and `"AttrB"` from the `test_cell` group for pin `"A"` in all the cells.

```
set_var -cell "A B" -pin "C D" test_cell_filter_attributes "attrA attrB"
```

Removes the attributes `"attrA"` and `"AttrB"` from the `test_cell` group for pins `"C"` and `"D"` in cells `"A"` and `"B"`.

timing_group_unateness

`<merge | separate>` Controls merging of positive/negative unate arcs into single non-unate group.
Default: `merge` (compatible with 3.2 behavior)

`separate` Liberate will ***not*** merge `pos_unate` and `neg_unate` arcs into a single non-unate timing group. This parameter applies to timing groups that are not default timing groups.

For control of merging for default groups, see the `-unateness` option of the set_default_group command.

`merge`	Merge two timing groups under the following conditions: ■ They have the same related_pin and `when` condition, the `timing_type` is `combinational`, and one is `positive_unate` and the other is `negative_unate`. (Default) ■ The `timing_type` will be changed to the appropriate edge. (For example, if the related_pin is a clock, then change the `timing_type` to `rising_edge` or `falling_edge` accordingly, otherwise, set it to `combinational`.) ■ The `timing_sense` will be removed if discard_timing_sense_after_merge is true, otherwise, it will be set to `non_unate`.

Setting this parameter to `separate` ***can adversely impact model generation for all define_arc/verbose flows.***

This parameter must be set before the char_library command is run.

tmpdir

`<string>`	Specify a temporary working directory. Default: `none`

This parameter specifies a temporary working directory for Liberate to store extsim run data.

You can specify a temporary working directory using the `tmpdir` Tcl variable or the `TMPDIR` environment variable. If both variables are set, the `tmpdir` Tcl variable takes precedence. Liberate follows the rules given below for creating temporary directories:

1. Use `tmpdir` (create dir if missing).
2. If `tmpdir` is not set use environment variable `TMPDIR` (create if missing).

3. Error out if either `tmpdir` or `TMPDIR` point to files.
4. If neither `tmpdir` nor `TMPDIR` are available use the current working directory (cwd).

This parameter must be set before `char_library`.

toggle_leakage_state

`< 0 | 1 >` — Toggle the clock before measuring state-dependent leakage on sequential cells.
Default: `0` (Off)

This parameter will ensure that the clock pin is toggled for a cycle to store a logic state in a sequential cell before measuring state-dependent leakage. Turn this on to disable fully-specified state-dependent leakage that includes output pins for all cells. That is, when this parameter is set to 1, the *when* specified in the leakage groups will not include the output pin state.

This parameter must be used before `char_library`.

Example

```
# Disable leakage conditions that depend on Q
set_var toggle_leakage_state 1
```

tran_tend_estimation_mode

`<0 | 1 | 2>` — Enables switching between different algorithms for estimating the transient simulation end time (`tran_tend`).
Default: `1`

- `0` — The estimation method uses 99.5% of the transition threshold which in some corner cases can take a long time to complete simulation, resulting in a longer simulation time for power simulations. Subsequently, the leakage power that is being subtracted from internal power calculations may become inaccurate.
- `1` — Estimate the `tran_tend` using the slew measurement threshold.

`2` Specifies to use `+errpreset=conservative` or user-defined options in `tran_tend` estimation. In addition, SKI is used with ski_meas_mode in `tran_tend` estimation.

tristate_disable_transition

`<0 | 1>` Requests Liberate to output full transition tables for the tristate disable arcs.
Default: `1` (copy delay to transition)

- `0` Liberate outputs a scalar value of 0 in the transition tables for `three_state_disable` arcs.
- `1` Liberate outputs full transition tables for `three_state_disable` arcs. The transition disable arc data is copied from the corresponding delay tables.

This parameter must be set before the write_library command is run.

tristate_pin_cap_always_on_res_mode

`<0 | 1>` Adjusts the behavior of adjust_tristate_load where the pin capacitance is very large due to pull-up and pull-down resistances that are always on.
Default: `1` (on)

- `0` Behavior of release LIBERATE 12.1 ISR1 and earlier.
- `1` Corrects the effect of `adjust_tristate_load` when pin capacitance is very large.

This parameter must be set before the char_library command is run.

unique_pin_data

`<0 | 1>` Requests unique pin data under all bundles and buses.
Default: `0`

- `0` Bundle and bus has one data table applied to the entire bus. Each `write_library` command may be generated with unique pin data by using the `-unique_pin_data` option of the `write_library` command.
- `1` The bus and bundle data is modeled with unique characterized data for each bundle and bus element.

  This setting does require that all bits of the bus are characterized and unique arc data exists. The setting is equivalent to the `-unique_pin_data` option of the `write_library` command.

This parameter is set for a specific cell, pin, related_pin, type by using the `set_var` command.

Example

To break-out per-pin data for power only, specify the following:

```
set_var -type power unique_pin_data true
```

This parameter must be set before the write_library command is run.

update_training_data

`<0 | 1>` Controls whether a training database should be updated with good tables found during the CCSN DC table checks.
Default: `0`

Note: A training database is an encrypted file that contains normalized DC tables.

`0` Do not update the training database.

`1` Update the training database.

This parameter must be set before the write_library and write_training_data commands are run.

To ensure that your training database is updated with good tables found during the CCSN DC table checks, set the update_training_data parameter to `1` before running the `write_library` command.

Example

Update the training database on the fly while checking the `write_library` data:

```
read_training_data $dbList
set_var update_training_data true
read_library $lib
write_library … -ccsn -fix_dc $lib
write_training_data $newDb
```

use_arcs_only_mode

`<-1 | 0 | 1>` Controls enabling and disabling of user_arcs_only mode.
Default: `-1`

The `use_arcs_only_mode` parameter takes precedence when it co-exists with the `-use_arcs_only` option of the char_library or `char_variation` command.

`-1` Follow the `-use_arcs_only` option set with the `char_library` or `char_variation` command.

0	Disable use_arcs_only mode even if the `-use_arcs_only` option is specified with the `char_library` or `char_variation` command.
1	Enable use_arcs_only mode. It functions the same way as does the `-use_arcs_only` option of the `char_library` or `char_variation` command.

This parameter can be set on a per-cell basis using the `-cell` option with the set_var command.

Example

```
# characterize INVD1 in regular mode, characterize INVD2 in user_arcs_only mode.
set_var -cell {INVD1} user_arcs_only_mode 0
char_library -user_arcs_only -extsim spectre -thread 1 -cells {INVD1 INVD2}
```

use_pid_tmpdir

<0 | 1>	Controls whether the process ID should be added to the tmp directory name. Default: 1
0	Does not adds the process ID to the tmp directory name.
1	Adds the process ID to the tmp directory name.

This parameter must be set before the char_library command is run.

Example

```
/tmp/<user>/altos.<pid>.0
/tmp/<user>/altos.<pid>.1
```

user_data_apply_after_ldb_processing

`<0 | 1>` Controls when the user data should be applied to the LDB.
Default: `0`

- `0` Applies the user data to the LDB before processing it and setting it up for `.lib` generation.
- `1` Applies the user data to the LDB after processing it. This is done to avoid overwriting the existing user data.

This parameter must be set before the write_library command is run.

user_data_attr_order

`<0 | 1>` Controls whether to keep the complex attributes in the exact order found in the user_data or to follow the default order used by Liberate when writing out a library.
Default: `1`

- `0` Liberate outputs attributes from the user_data file (see write_library) when writing out a library.
- `1` Liberate outputs the complex attributes in the exact order found in the user_data file.

This parameter must be set before the write_library command is run.

user_data_ignore

`"test_cell"` Specifies attributes or groups to ignore from the file identified with the write_library `-user_data` command. The supported value for this parameter is `"test_cell"`.
Default: `1`

When this parameter is set, you do not need to remove the `test_cell` from the `user_data` file to ensure that the `test_cell` is removed or ignored from the output library. Using this setting allows the automatic `test_cell` generation to be utilized (see auto_test_cell).

This parameter must be set before the write_library command is run.

Example

```
set_var user_data_ignore "test_cell"
```

user_data_override

`{attribute_list}`	A list of attributes that will override characterized values during `write_library -user_data`. Default: `{area function three_state state_function}`

This parameter allows the user to control which entries in the user data file will override the corresponding data in the `ldb` when using the `-user_data` option of the `write_library` command. It specifies a list of items which, if found in the user data file, will replace the value in the ldb. This list will append values to the built-in list of overrides.

Currently, the following elements are valid for the override list:

- `capacitance`
- `fall_capacitance`
- `rise_capacitance`
- `fall_capacitance_range`
- `rise_capacitance_range`
- `max_capacitance`
- `min_pulse_width` (for both `min_pulse_width_high` and `min_pulse_width_low`)
- `min_pulse_width_high`
- `min_pulse_width_low`
- `cell_leakage_power`
- `voltage_map`
- `pg_pin`
- `min_transition`
- `max_transition`

The built-in default list of attribute overrides currently includes:

```
    {area function three_state state_function}
```

This parameter must be set after the char_library command is run.

Example

```
set_var user_data_override \
    {max_capacitance cell_leakage_power}
```

user_data_quote_attributes

`{ list }`	Specifies a list of attributes whose values need to have double quotes surrounding them. Default: `{ }` (No attributes surrounded with quotes.)

This parameter must be set before the write_library command is run.

user_data_quote_simple_attr

`<0 | 1>`	Specifies whether simple, non-numeric attributes from the user data are wrapped in quotes. Default: `1`

When **user_data_quote_simple_attr** is set to 1 and **user_data_attr_order** is set to 1, simple, non-numeric attributes from the user data are wrapped in quotes. Set this parameter to 0 for 2.5p2 and prior behavior.

This parameter must be set before the char_library command is run.

vector_check_initial_mode

`< 0 | 1 >` Controls whether vector integrity check is needed before characterization.
Default: 0 (disabled)

- `0` Specifies not to check the integrity of any vector.
- `1` Adds a check to avoid generation of vectors with inconsistent internal nodes.

This parameter must be set before the char_library command is run.

vector_check_mode

`< 0 | 1 | 2 | 3>` Controls the algorithm used to determine the cell behavior for a particular vector. This parameter helps in rare cases where Liberate fails to find an arc which is known to exist. This may occur in those cases where there is an internal race condition.
Default and recommended:`3`

- `0` Uses internal logic simulator for checking vector validity, but ignores internal delay. The results of this setting are less precise than 3.
- `1` Determines the cell behavior for a particular vector. It is used for backward compatibility.
- `2` Uses characterization simulator for checking vector validity. The results generated from this setting are more accurate than `3`, but Liberate runs much slower.

  Note: The characterization simulator is specified by using the char_library `-extsim <simulator>` command.
- `3` Uses the internal logic simulator for checking vector validity.

This parameter must be set before the char_library command is run.

vector_estimate_dump

`< 0 | 1 >`

Prints the result of the estimation decks for use in vector debugging.
Default (and recommended): `0` (Off)

- `0` — Specifies not to print vector estimate information.
- `1` — Prints vector estimate information intended for template debugging purposes.

For each `define_arc` with more than one possible vector, Liberate prints a list of vectors along with the estimated delay, power, and so on. It also prints which vectors are chosen for full simulation. This only includes vectors for which an estimate is done. If `define_arc` has only one vector, the results of the full simulation should be reviewed.

This parameter must be set before the char_library command is run.

vector_side_input

`< 0 | 1 >`

Controls whether the unknown side inputs should be set to 0 or 1.
Default: `-1` (expand all X)

This parameter can be used to force side inputs for which the values are not specified in a "when" condition or a vector to all zeros or all ones. By default, the tool automatically decides what the side inputs should be set to. This includes leaving them floating. It is recommended to always set all side inputs.

- `0` — Sets the unknown side inputs to 0:X.
- `1` — Sets the unknown side inputs to 1:X.

This parameter must be set before the char_library command is run.

verilog_cg_filter_edge

`<0 | 1>` Enable filtering of extra timing constraints in Liberty and Verilog.
Default: `1`

If this parameter is set to 1, extra timing constraints will be filtered out of Liberty and Verilog for cells that contain the attribute `clock_gating_integrated_cell`, according to the following table:

clock_gating_integrated_cell =	posedge	negedge
Filter these out:	min_pulse_width_**high** constraint_**high**	min_pulse_width_**low** constraint_**low**

Set this parameter to 0 to disable this filter.

This parameter must be set before the write_library or write_verilog command is run.

verilog_use_internal_as_inout

`<true | false>` Controls the behavior of the write_verilog command for treating internal pins in the input library (direction : internal).
Default: `false`

- `true` Treats the internal pins as inout pins in the Verilog file.

  This type of Verilog file is useful for SDF backannotation using the `validate_sdf` command of Liberate LV.
- `false` Disables the functionality.

voltage_map

`< 0 | 1 | 2>` Generate a voltage map in the output library.
Default: `1`

`0`	Disable generation of `voltage_map` attributes in the library.
`1`	Generate `voltage_map` attributes at the library level along with `pg_pin` groups for each cell and `related_power_pin` / `related_ground_pin` attributes for each pin. (*Liberty 2006.06 syntax*)
`2`	Generate a `power_supply` group along with `rail_connection` attributes for each cell and `input_signal_level` / `output_signal_level` attributes for each pin. (*pre-Liberty 2006.06 syntax*)

This parameter can be used to generate a set of voltage map attributes in the output library. This should be used when multiple voltage levels are used within a cell, for example, a level shifter.

If `voltage_map` is set to `1` or `2` an *operating_conditions* group will be generated in the output library with the appropriate *process, temperature, voltage* and *signal_level* attributes. The name of the operating group defaults to "*PVT_<process>_<voltage>V_<temp>C"* (with decimal points (.) being replaced by the character P). This can be overwritten by providing a named *operating_conditions* group as `user_data` to the `write_library` command. The *process* attribute within the *operating_conditions* group will also be overwritten by the value in the user_data file.

Note: If the `char_library -ccsp` option is enabled, this parameter will be overridden. A CCS-Power library requires the *voltage_map, related_power_pin, and related_ground_pin* attributes and the *pg_pin* and *power_supply* groups.

Liberate will support multiple power formats in the following way:

2005.12 format:

```
set_var voltage_map 2
char_library -ccsp
write_library
```

2006.06/9 format:

```
set_var voltage_map 1
char_library -ccsp
write_library
```

CCSP format:

```
char_library -ccsp
write_library -ccsp
```

This parameter must be set to a non-zero value before the char_library command is run.

This parameter can be set to 0 after the read_ldb command and before the first write_library command is run.

Example

```
# Use multiple supply voltages
set_operating_condition -voltage 1.08 -temp 25
set_vdd VDDL 0.84
set_gnd VSSL 0.02
set_var voltage_map 1
```

The following constructs will appear in the library :

```
default_operating_conditions : PVT_TT_1P08V_25C;
voltage_map (VDD_VSS, 1.08);
voltage_map (VDDL_VSSL, 0.82);
.
cell (BUF_1) {
    pg_pin (PP1) {
    pg_type : primary_power;
    voltage_name : VDD;
    }
    pg_pin (GP1) {
    pg_type : primary_ground;
    voltage_name : VSS;
    }
    pg_pin (PP2) {
    pg_type : primary_power;
    voltage_name : VDDL;
    }
    pg_pin (GP2) {
    pg_type : primary_ground;
    voltage_name : VSSL;
    }
    pin (X) {
    direction : output;
    function : "A";
    max_capacitance : 0.089809;
    related_ground_pin : GP2;
    related_power_pin : PP2;
    timing () {
        ....
```

```
    }
    }
}
```

vsrc_slope_mode

<0 | 1>	Improved Alspice PWL breakpoint handling. Default: 1

This allows for improved Alspice PWL breakpoint handling. The default and recommended values are 1. (Note: 0 is not recommended; this is only for compatibility with releases prior to 3.1.

This parameter must be set before the char_library command is run.

waveform_report

<0 | 2>	Enable saving of driver waveform. Default: 2 (The driver input waveform will be saved into the .vdb and the .ldb file.)

Set this control parameter to 0 to disable the saving of the driver input waveform into the .vdb file (see write_vdb). Do not use this parameter if the write_library -driver_waveform option is to be used.

This parameter must be set before the char_library and/or write_vdb command is run.

when_exclude

`<list_of_pin_name_patterns>`

Specifies a list of pin name patterns. The pin names that match one of the list elements are excluded from the automatically generated `when` conditions.
Default: `nil`

Scope: Can be restricted by `-cell`, `-pin`, or `-related_pin`.

Note: The `when` conditions specified with the define_arc command are not affected.

This parameter must be set before the char_library command is run.

Example

For more accurate power calculations, if multibit cells require the ability to control data `when` conditions, you can use commands such as following to put master-switching power on data and limit clock to insertion power:

```
set_var -type hidden -pin CK power_vector_selection_mode min
set_var -type hidden -pin D* power_vector_selection_mode max
set_var -pin D* when_exclude CK
```

In this example, when the D pin switches, master-switching power is counted regardless of whether it actually occurred when D switched or when CK switched. To make this work, hidden D does not depend on the CK state and counts the maximum over the two CK states, while CK counts the minimum over the four D/Q states.

wnflag

`<0 | 1>`

Instructs Liberate to use W/nf for model selection. If the user has `.param wnflag=0`, they must set the `wnflag` parameter to `0` in their Tcl run script to inform Liberate to use W *instead* of W/nf to select model binning.
Default: `1`

`0` Use W for model selection.

`1` Use W/nf ratio for model selection.

This parameter must be set before the read_spice command is run.

Example

```
set_var wnflag 0
```

worst_vector_reltol

<value>

Specifies a tolerance value to prevent multiple vectors from being simulated if the difference between two vectors is not significant. Ensure that the specified value is >0.0, but <=1.
Default: 0.02

Note: This parameter is used when the worst_vector_selection_mode parameter is set to 4.

If the worst-case vector at a point is within the specified tolerance of another vector and that other vector is worst-case at a different point, then only the other vector needs to be fully simulated.

This parameter must be set before the char_library command is run.

worst_vector_selection_mode

<0 | 1 | 4>

Specifies the worst-case vector selection mode that Liberate must use.
Default: 4

- 0 — Simulate all vectors and use point-by-point worst case.
- 1 — Select a single worst-case vector and simulate the center table point of each vector to determine which is worst.
- 4 — Select up to four worst-case vectors and simulate the table corner points to determine which vectors are worst.

This parameter must be set before the char_library command is run.

write_datasheet_mpw_use_table_style

`<0 | 1>` Controls whether the MPW constraints should follow other constraints and conform to `write_datasheet -table_style`, which specifies where to create the datasheet tables from—first-mid-last or min-avg-last.
Default: `1`

- `0` This setting is for backward compatibility with the LIBERATE 19.2 and prior releases. When the parameter is set to `0`, the MPW constraints are displayed in a simple table and include only maximum values.
- `1` Ensures that the MPW constraints follow other constraints and conform to `write_datasheet -table_style`.

This parameter must be set before the write_datasheet command is run.

write_library_allow_switching_and_hidden_power

`<0 | 1>` Control whether to output both hidden power and switching power groups at the same time.
Default: `0`

For inout pins, such as PAD pin, that have both switching power and hidden power groups, only switching power groups are output by default. Use this parameter if both need to be output.

- `0` Outputs only the switching power groups for an IO pin.
- `1` Outputs both hidden power and switching power groups for an IO pin.

This parameter must be set before the char_library command is run.

write_library_is_unbuffered

`<0 | 1>` Controls the output of `is_unbuffered` attribute in .lib.
Default: `0`

`0`	Outputs the Liberty `is_unbuffered` attribute for CCSN arcs.
`1`	Outputs the Liberty `is_unbuffered` attribute for non-CCSN arcs.

This parameter must be set before the char_library command is run.

write_library_mode

`<0 | 1>` Enables faster method for writing a library in an arc packet flow. Default: `0`

`0`	Writes a library in the server containing all characterized cells
`1`	The `write_library` command is distributed 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 in the server. This results in the library creation being distributed across the clients.

This flow requires that the `write_library` command executes in the server and in the client. Any existing script that protects the `write_library` command so it cannot execute in the client (see `packet_slave_cells` and `ALAPI_active_cell`) must be updated. If the `write_library` command does not execute in the client, the library is written on the server as when `write_library_mode` is set to `0`.

This parameter must be set before the write_library command is run. Writing a library with a setting of `1` followed by writing a library with a setting of `0` is not supported.

write_library_use_read_library_attr

<0 | 1>

Keeps the attributes in the output library the same as in the original library when this parameter is used in a flow where the read_library command is followed by the write_library command.
Default: 1

- 0 — Used for backward compatibility with the Liberate 16.1 ISR4 release. When this value is used, Liberate might override the max_transition and min_transition attributes depending on the Tcl settings. For example, if the write_min_transition_attr parameter is set to 1, the min_transition attribute will be overridden or added with the smallest transition index value.
- 1 — Ensures that attributes including max_transition and min_transition from the original library will be written to the output library. The attributes will not be overridden or added in the read_library and then the write_library flow.

This parameter must be set before the write_library command is run.

write_library_sync_ldb

<0 | 1>

Forces Liberate to read the LDB on disk prior to writing.
Default: 0

- 0 — Liberate will not read the LDB prior to writing the library.
- 1 — Forces Liberate to read the LDB on disk prior to writing the library. This setting is used to address possible precision issues between the char_library and read_ldb/write_library flow. (See also write_library -sync_ldb.)

This parameter must be set before the write_library command is run.

write_logic_function

`<0 | 1>`

Enables or disables writing of the *logic function* when using the *Inside View* algorithm. Disabling this feature is useful when Liberate has difficulty generating the correct function for complex cells. In that case, the user can provide the function in the `user_data` file to write_library, which will *override* any Liberate-generated function. Using `user_data` can also reduce run time.
Default: `1` (Enable)

- `0` Disables logic function determination when using the *Inside View* algorithm.
- `1` Enables writing of the logic function and tristate attributes.

This parameter must be set before the char_library command is run.

write_logic_function_failure_action

`<none | 0 | warn | 1 | clean | 2 | error | 3>`

Sets the conditions for writing a function attribute to the library.
Default: `0`

- `none | 0` If the function attribute resolved by Liberate is `0` or `1` and the cell is not a tie cell, write that function attribute to library without any message.
- `warn | 1` If the function attribute resolved by Liberate is `0` or `1` and the cell is not a tie cell, write that function attribute to library then output a warning message.
- `clean | 2` If the function attribute resolved by Liberate is `0` or `1` and the cell is not a tie cell, will not write that function attribute to library and output a warning message.
- `error | 3` If the function attribute resolved by Liberate is `0` or `1` and the cell is not a tie cell, write that function attribute to library, then output an error message and mark that cell as failed.

This parameter must be set before the char_library command is run.

write_logic_function_group_at_end

`<0 | 1>` Specifies the desired position for the function groups in the library. It instructs Liberate where to place the function groups (ff, latch, state_table) in each constrained cell in the library.
Default: `1` (ff/latch after the pins and state_table before the pins).

- `0` Places the function groups before the first pin in the cell.
- `1` Places the ff and latch function groups after the last pin in the cell and the state_table (see write_logic_function_mode) before the pins.

This parameter must be set before the char_library command is run.

write_logic_function_mode

`<0 | 1 | 2>` Enables an alternate algorithm for determining logic function, designed to minimize X-propagation.
Default: `2` (on)

Note: This algorithm does *not* guarantee matching the X-propagation behavior of all circuits.

- `0` Standard algorithm for generating logic function.
- `1` Alternate algorithm for generating logic function.
- 2 Alternate algorithm for generating logic function with redundant terms to reduce X-propagation.

This parameter must be set before the write_library command is run.

write_logic_function_statetable_limit

`<value>` Sets the maximum number of sequential elements allowed when generating a statetable format function when the write_logic_function_statetable_mode parameter is enabled.
Default: `16`

This parameter must be set before the char_library command is run.

write_logic_function_statetable_mode

`<0 | 1 | 2>` Enables the generation of a cell function using statetable format in the output library for complex cells.
Default: `0`

Note: This algorithm requires mega mode to be enabled (see the mega_enable parameter and the define_cell `-type` command). The `-io` option must not be used with the char_library command because it disables all the *Inside View* algorithm including the mega mode analysis.

- `0` Do not generate statetable format functions.
- `1` Generate statetable format functions for cells with more than one sequential element.
- 2 Generate statetable format functions for all cells with sequential elements.

This parameter must be set before the char_library command is run.

write_min_transition_attr

`<0 | 1>` Writes the `min_transition` pin attribute to the library.
Default: `0`

- `0` Do not output the `min_transition` pin attribute.
- `1` Output the `min_transition` pin attribute.

This parameter must be set before the write_library command is run.

write_template_ccsn_arc_include_timing_type

`<0 | 1>` Controls whether CCSN user-defined arc should check the `timing_type` if you specify `timing_type` in the corresponding CCSN define_arc command.
Default: `0`

- `0` Do not write the following attribute in the CCSN arc:

```
-attribute {arc_based_ccsn_timing_type preset/
     clear...}
```

`1` Write the following attribute in the CCSN arc:

```
-attribute {arc_based_ccsn_timing_type preset/
     clear...}
```

For arc-based CCSN, the write_template `-ccsn` command might write out duplicated arcs. Use this setting to avoid it.

This parameter must be set before the write_template command is run.

Example

```
define_arc \
    -type ccsn_last \
    -when {(CK * !(SD))} \
    -pin Q \
    -related_pin RD \
    -attribute {arc_based_ccsn_timing_type clear} \
    -ccsn_stage arc \
    cellname
set_var write_template_ccsn_arc_include_timing_type 1
```

write_template_ccsn_default_arc_include_is_inverting_attr

`<0 | 1>` Controls whether or not to write the attributes defined with the `define_arc -attribute` command for the CCSN default group.
Default: 1

`0` Do not write `define_arc -attribute {is_inverting}` for CCSN default group.

`1` When this value is set for `write_template_ccsn_default_arc_include_is_inverting_attr`, and, if `write_template -ccsn` is set, write `define_arc -attribute {is_inverting}` for CCSN default group.

This parameter must be set before the write_template command is run.

write_template_dc_current

`<0 | 1>` Controls whether or not the write_template command will write define_arc commands of type `dc_current` into the template file for power switch cells.

- `0` Do not write `define_arc` commands for `dc_current` of switch cells into the template file.

  Note: This setting is for backward compatibility with the LIBERATE 17.1 ISR2 and prior releases that did not support the output of `dc_current` related `define_arc` commands.

- `1` Write `define_arc` commands into the template for `dc_current` arcs of power switch cells when the `-verbose` option is specified with the `write_template` command.

This parameter must be set before the `write_template` command is run.

write_template_pvt_filename_pin_supply_name_mode

`<0 | 1>` Controls setting of the mode that should be used for generating the `set_pin_vdd`/`set_pin_gnd` PVT template.
Default: `1`

- `0` Specifies that while using `write_template -pvt_filename` for generating PVT template of `set_pin_vdd`/`set_pin_gnd -supply_name`, the exact supply name that is set in `set_vdd`/`set_gnd -name_map` is output for all the cells.
- `1` Specifies the mode for `write_template -pvt_filename` flow to fix CCSN `name_map` mismatch issue that is caused by the `set_vdd`/`set_gnd -name_map` settings.

  In this mode, the `-name_map` settings from the `set_vdd` and `set_gnd` commands are disabled from all the cells while generating the `set_pin_vdd`/`set_pin_gnd` PVT template.

This parameter must be set before the write_template command is run.

6

Library Comparisons

This chapter describes the Liberate utilities for comparing libraries.

Liberate provides ways to compare two libraries. This is useful, for example, to understand how new SPICE models may change library characteristics such as leakage power. The comparison can also be used to compare Liberate-generated libraries against existing libraries.

Both textual and graphical comparisons can be generated. The compare_library command can be used to generate a text comparison report highlighting outliers that exceed the defined absolute and relative tolerances. Use the `-gui` option with `compare_library` to generate an intermediate file for graphical comparisons. The graphical comparison utility is called `lcplot`.

lcplot

`<gui comparison file>`	Specifies the file generated by the `compare_library -gui` command option.

The `lcplot` graphical comparison plots make it easy to pin-point library entities that have significant differences, or to confirm expected trends such as slower delays when comparing a library generated at a slow corner versus a fast corner.

Panel Buttons

Fit X	Expand the current graph in the X range to full scale while keeping the Y range fixed.
Fit Y	Expand the current graph in the Y range to full scale while keeping the X range fixed.
Fit All	Expand the current graph in both the X and Y ranges to full scale.
Log - X	Change the X-axis in an X/Y style graph into a logarithmic scale.
Log - Y	Chang the Y-axis in an X/Y style graph into a logarithmic scale.
Timing	Select only Timing Data to display (delay, setup, hold, recovery, removal and trans).
Power	Select only Power Data to display.
Leakage	Select only Leakage Data to display.
Capacitance	Select only Capacitance Data to display (capacitance, fall_capacitance and rise_capacitance).
Style	Select the style of the graph to display. More details can be found below.
Cell Sel	Brings up 'Cell Selection' form to select cells to display
Data Sel	Brings up 'Data Selection' form to select data type to display
Direction	Brings up 'Direction Selection' form to select data toggle direction (rise, fall)
Close	Closes the lcplot window.

Drop-Down Menus

File -> Print	Brings up Print form to print to file (postscript format) or printer in gray scale or color.
File -> Exit	Close lcplot
View -> ZoomOut 2	Zoom out by 2X (or, click the Right Mouse Button in graphic display window to zoom out by 2X)
View -> ZoomIn 2	Zoom in by 2X (or, use the Left Mouse Button in graphic display window to select a box to automatically zoom into).

Redraw	Redraw the window to clean up any cursor ghosts.
Help -> About	Display Version and Copyright notice.

Plot Styles for Graphical Library Comparisons

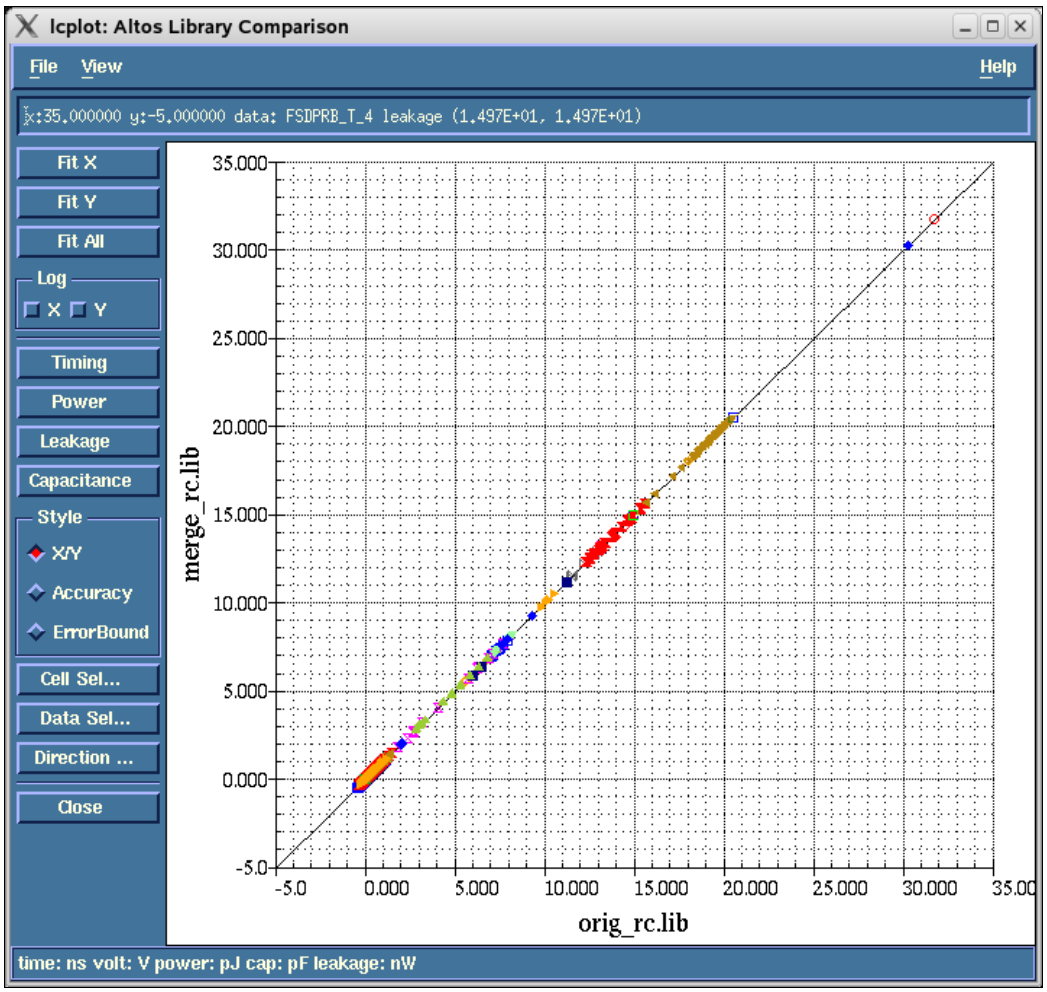
The following three styles of plots are available for graphical library comparisons: X/Y, Accuracy, and Errorbound.

Style: X/Y

The graphical X/Y library comparison shows a scatter plot where the values from the reference library are given on the X-axis and the values from the comparison library are given on the Y-axis. When the values in the two libraries match, the plotted data points fall on the 45-degree axis.

Note: By default, the X/Y library comparison plot type displays all four data types: Timing, Power, Leakage, and Capacitance.

An example of X/Y library comparison in the *lcplot* window is shown below:

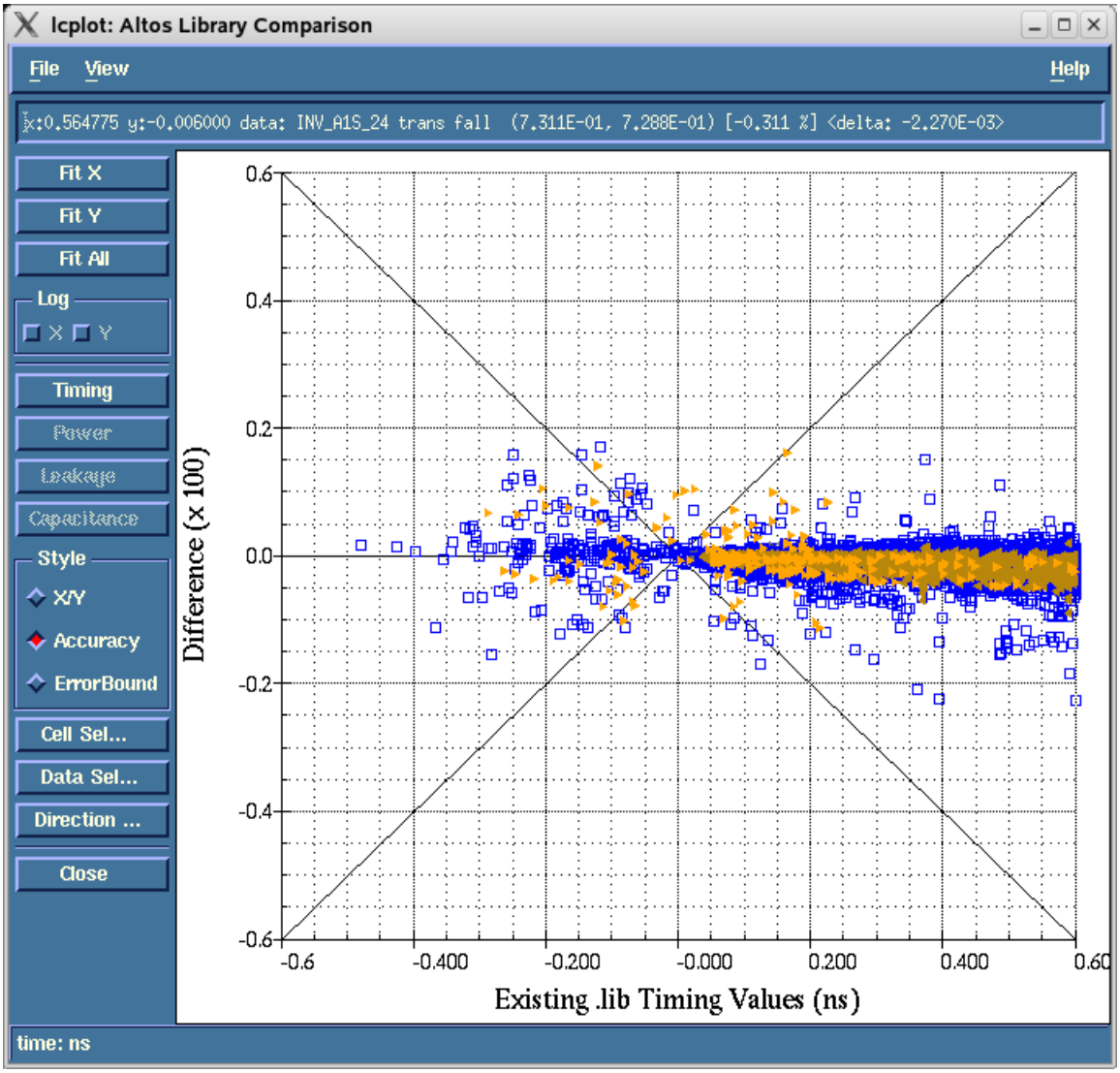


Style: Accuracy

The following graphical library comparison shows an Accuracy plot where the existing value in the reference library is given on the X-axis and the absolute difference in values (scaled up by 100) between the comparison library and the reference library are given on the Y-axis. This plot also includes a +45 degree and a -45 degree axis. These two axes form four triangular regions. The left and right regions represent the data points that fall within 1%. The upper and lower triangular regions represent the data points that are greater than 1% of difference. This plot is useful in determining which data points are greater than 1%. When the mouse is positioned directly over a data point, the actual data from both libraries will be displayed in the frame directly above the graph.

Note: This plot style only applies to timing comparisons.

An example of accuracy library comparison in the *lcplot* window is shown below:



Style: ErrorBound

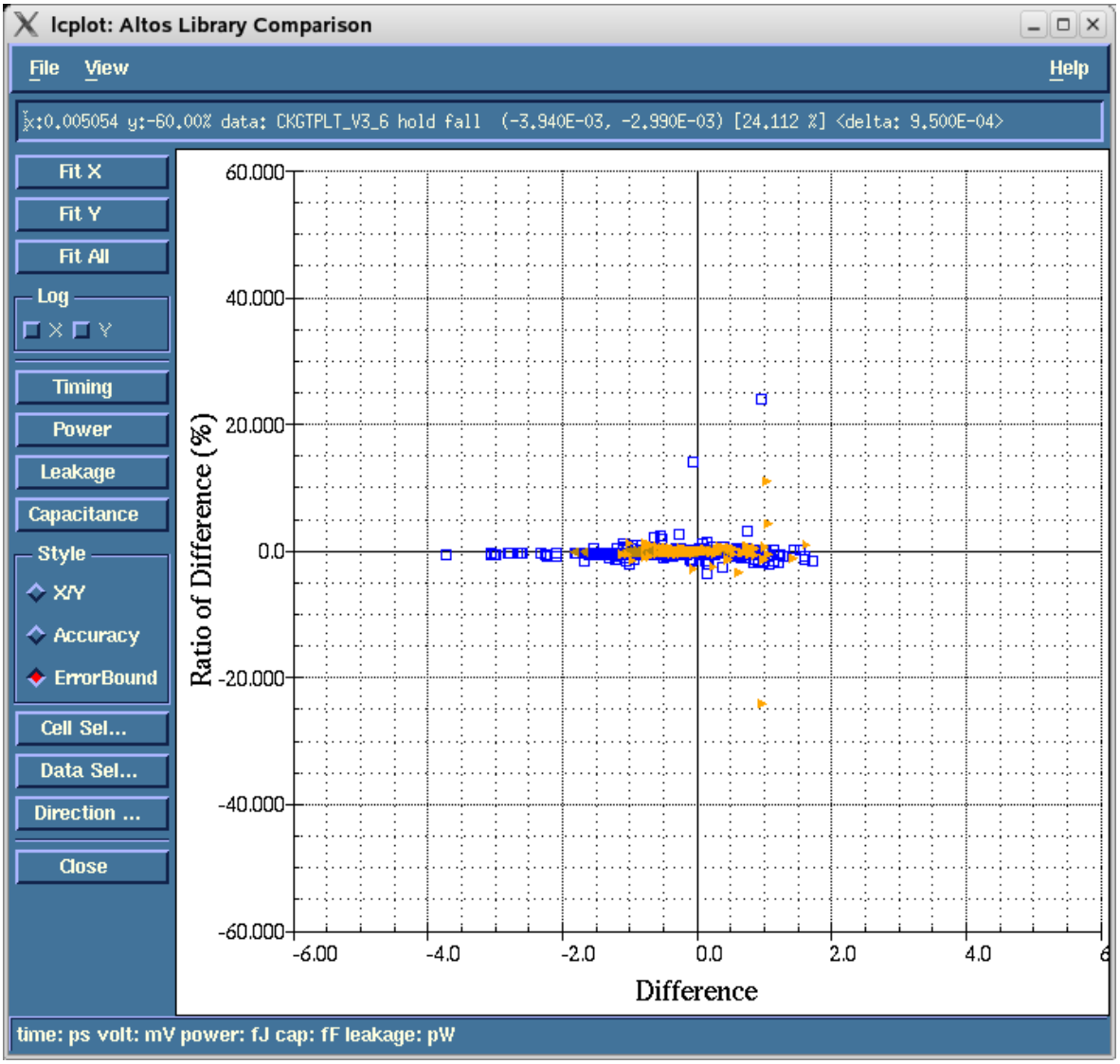
The following graphical library comparison shows an ErrorBound plot where the absolute difference in values between the comparison library and the reference library is given on the X-axis, and the Difference Ratio `((compVal-origVal)/origVal)*100` of the comparison library to the reference library is given on the Y-axis.

- If the data point on the graph falls close to the X-axis zero reference (X=0), then the absolute error is small so the relative error can be ignored, even if it is large.

- If the data point falls close to the Y-axis zero reference (Y=0), the ratio of difference is small and even if the absolute difference is large, this difference can be ignored.
- When data points do not fall near either of the zero reference axes, the difference may be significant and should be reviewed.
- When the mouse is positioned directly over a data point, the actual data from both libraries will be displayed in the frame directly above the graph.

Note: By default, this plot type will display all four data types: Timing, Power, Leakage and Capacitance.

An example of ErrorBound library comparison in the *lcplot* window is shown below:



Data Selection

There are 2 ways to select data: Cell Type, and Data Type.

Selecting a Cell Type

1. To select the cells to compare, click the *Cell Select* button.

   This displays a selection window that lists all the cells in the library, as shown below.

2. To select a cell from the list, click the cell name and then click the *Apply* button.
3. To select multiple cells, press the `Shift` key, select the cell names, and click *Apply*.
4. To add cells to the selected set, press the `Ctrl` key, select the cell names, and click *Apply*.
5. To select all cells, click the *All* button and then click the *Apply* button.
6. To deselect all cells, click the *None* button and then click the *Apply* button.
7. To exit the cell selection window, click the *Close* button.

Selecting a Data Type

1. To select the type of library data to compare, click the *Data Select* button.

This displays a selection window that lists all the available data types, as shown below.

2. Select the required data type from the list and click the *Apply* button.
3. To compare multiple data types simultaneously, press the `Shift` key, select the required data types, and then click *Apply*.

   The unit for each displayed value is defined by the reference library.

The following data types are available for comparison (assuming they are present in the reference library).

trans	Transition times
capacitance	Input pin capacitance
power	Switching and hidden power
leakage	Leakage power
fall_capacitance	Pin capacitance for falling transitions
rise_capacitance	Pin capacitance for rising transitions
delay	Transition delays
recovery	Recovery time constraints
hold	Hold time constraints
setup	Setup time constraints
removal	Removal time constraints

As the cursor moves across the plot window, the corresponding data represented by each comparison point is highlighted in the window pane header. This shows both the reference and comparison values, and their percentage difference.

To zoom into the selected area (when the cursor is in the plot window), click and drag the cursor to form a rectangle over the plot area. A single right-click zooms out the plot by 2X (when the cursor is in the plot window). The *fit_x*, *fit_y*, and *fit_all* buttons can be used to fit the data within the window after zooming.

7

Performing Characterization using Liberate

This chapter details how Liberate performs various characterization tasks. Liberate characterizes the following constructs:

Delay Models

- Non-linear Delay Model (NLDM)
- Composite Current Source (CCS) delay model
- Effective Current Source Model (ECSM)

Pin Capacitance

- Non-linear Delay Model (NLDM)
- CCS Receiver capacitance
- ECSM capacitance

Timing Constraints

- Setup & Hold
- Recovery & Removal
- Minimum Pulse Width

Power Models

- Leakage Power
- Switching & Hidden Power
- Power Subtraction
- Power Validation

- Common Usage Models for Power and Leakage

Signal Integrity Models

- Steady State Current
- Noise Immunity Curve
- Hyperbolic Input Noise
- DC Noise Margin

Composite Current Source Noise (CCSN) Models

- CCSN DC current
- CCSN Output Voltage
- CCSN Miller Capacitance
- CCSN Propagated Noise

Electromigration Models

- Electromigration Maximum Toggle Rate

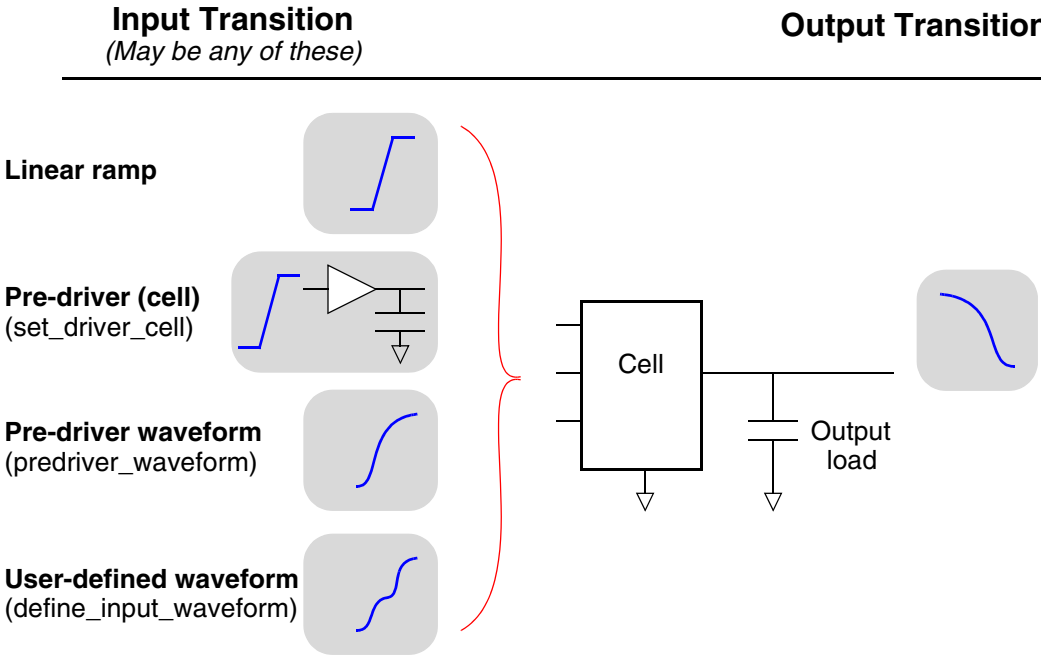
Delay Models

Non-linear Delay Model

The Non-linear Delay Model (NLDM) delay model is characterized by measuring the delay and output transition when simulating a given range of different combinations of input transitions and output loads. For defining the input waveform shape. you can use one of the following methods given in the order of precedence from highest to lowest:

- User defined input waveform (highest precedence):
  - Method 1 (higher precedence): normalized_driver_waveform parameter
  - Method 2: define_input_waveform command
- Analytical driver waveform: predriver_waveform parameter
- Output of a pre-driver: set_driver_cell command
- Linear ramp from the defined template (lowest precedence)

The input pin is triggered by a ramp or a smooth waveform (output of pre-driver). The output pin load is a simple capacitance.



Circuit used for delay and output transition measurements

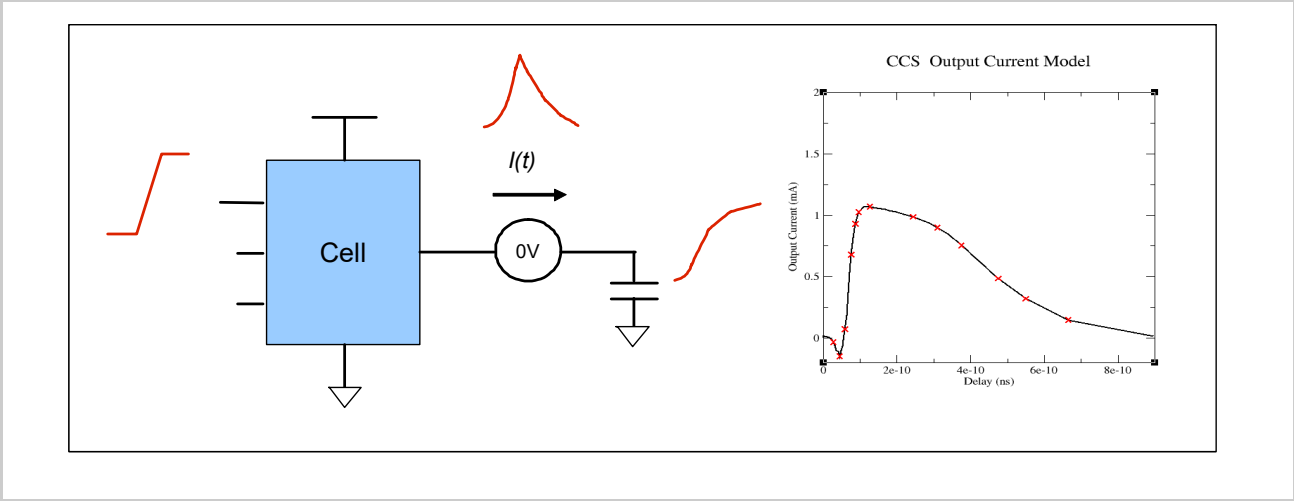
Delay is measured from the input crossing the delay measurement point (delay_inp_rise/ delay_inp_fall) to the output crossing the delay measurement point (delay_out_rise/ delay_out_fall, default 50% of supply). Output transition time is measured from the lower-to-upper slew measurement points (default 30% to 70% of supply) for rising transitions and from the upper-to-lower measurement points (default 70% to 30% of supply) for falling transitions. The slew measurement points can be reset using the `measure_slew_*` parameters.

All timing arcs are automatically and exhaustively characterized. For each arc, every side input combination is considered. If an arc has conditional delays, then all conditional tables will be generated, as well as the unconditional table which contains the worst-case results for each table entry.

Input transition and output load indexes may be specified uniquely for each arc, cell or group of cells using the define_index and define_template commands.

Composite Current Source

The CCS timing model includes output current characterization using a similar circuit to the circuit used in NLDM delay and transition measurements. In addition, a voltage source is attached to the output pin before the load capacitor to measure current flowing out of the pin.



Circuit used for CCS Driver Model

The current waveform is automatically simplified into a number of time/value pairs, which accurately model the original current waveform. Liberate will verify the accuracy of the simplified waveform and make the necessary adjustments to create the optimal number of points that maintain the required accuracy.

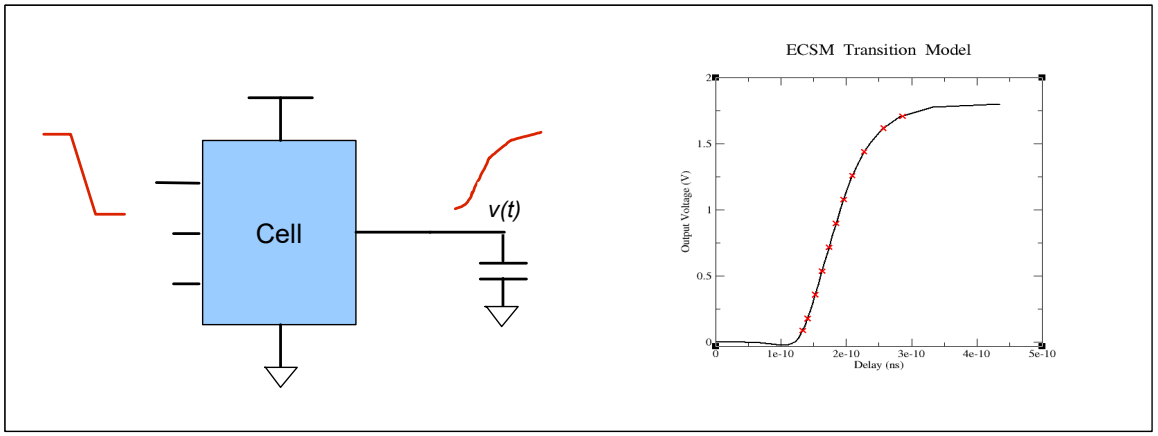
Dynamic selection of points ensures accuracy with minimal data size. The current waveform is measured such that the final voltage reaches within 0.1% of Vdd or Ground, and the integrated Voltage is calculated within 0.1ps of NLDM. CCS receiver capacitance is measure for two values above and below the delay threshold. No performance overhead is incurred over NLDM model characterization, as both models are calculated in the same simulation.

Liberate samples currents, then integrates them back to voltage. A comparison is done between this "reconstructed" voltage and the original voltage. Liberate will stop choosing more points when any one of the following criteria is satisfied. Liberate will select up to ccs_max_pts or until the ccs_abs_tol or the ccsp_rel_tol has been met.

Note: Increasing the number of points and lowering the `ccs_abs_tol/rel_tol` will increase the library size. It has no impact on characterization run time.

Effective Current Source Model (ECSM)

The ECSM delay model includes output voltage characterization using the same circuit as in NLDM delay and transition measurements. In addition, many more time points are captured during the output transition. The voltage thresholds for these time points are set by the ECSM template (**define_template -type ecsm**). By default Liberate measures 11 points.

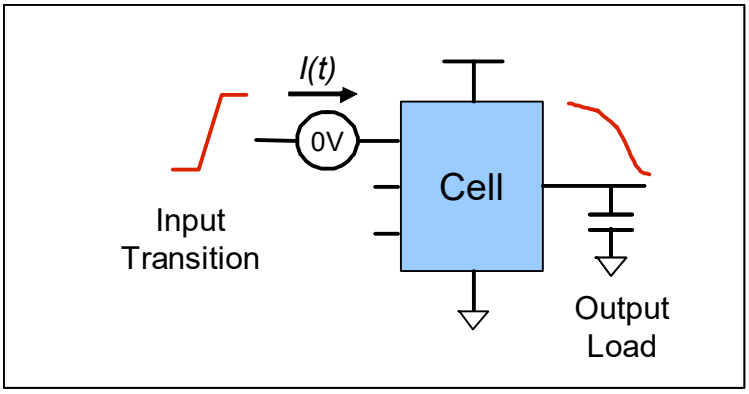


Circuit used for ECSM Driver Model

Pin Capacitance

Non-linear Delay Model Capacitance

Pin capacitance is characterized by measuring the current injected into the input pin over a time period, that is, charge divided by the voltage change on the input pin over that same duration of time. See circuit below:



Circuit used for pin capacitance measurement

Pin capacitance is measured at the same time as delay and transition. The reported capacitance is the worst capacitance value, as measured from each timing arc originating from the pin and for each table entry in the arc.

For rising transitions, pin capacitance is measured when the input transitions from the `measure_cap_lower_rise` threshold (default ground) to the `measure_cap_upper_rise` threshold (default 50% of supply). For falling transitions, pin capacitance is measured when the input transitions from the `measure_cap_upper_fall` threshold (default supply) to the `measure_cap_lower_fall` threshold (default 50% of supply).

Note: If the input pin always has an on parasitic pull-up or pull-down resistance, the pin capacitance measurement reports a false (large) pin capacitance. In this case, the `user_data_override` parameter can be used in combination with the `-user_data` option of the `write_library` command to override the characterized pin capacitance with a specified value.

Composite Current Source Receiver Capacitance

Liberate measures the Composite Current Source (CCS) receiver capacitance as per the CCS Timing Characterization Guidelines. The guidelines require measuring the capacitance as segments of the input transition. There are following two supported formats:

- **C1C2**: This format is a 2-piece segment that uses `C1` and `C2`, where `C1` represents the capacitance from the start to the delay threshold and `C2` represents the capacitance from the delay measurement threshold to the second slew measurement threshold. Both segments reflect the local slews.
- **C1CN**: This format is a multi-piece segmented input receiver capacitance model where the input capacitance is split into many segments of the input transition (See the set_receiver_cap_thresholds command and the ccs_cap_enhancement_format_mode parameter for more details).

See the *Liberty Timing Characterization Guidelines* for more details about CCS Receiver Capacitance modeling.

Effective Current Source Model Capacitance

Effective Current Source Model (ECSM) capacitance characterization utilizes the same measurement circuit as the NLDM model. In ECSM capacitance measurement, the

capacitance values for all input-slew/output-load combinations are reported in a table, not just the worst-case.

For rising transitions, current is measured from the input at 0 Volts to the value listed in the "threshold_pct" attribute in the Liberty model. For one-piece models, this is usually 0-50% of the supply. For n-piece models, the measurements start at 0 Volts and continue to the threshold_pct listed in each table. Capacitance measurements for falling transitions start at the upper rail and end at the threshold_pct value. The actual capacitance values are determined by integrating the current over the range of the curve and dividing that by the change in voltage.

Timing Constraints

Liberate automatically determines timing constraints on sequential cells. Liberate does not require the logic function or stimulus to be specified, merely that each pin be classified into one of the following types using the **define_cell** command: **input**, **output**, **bidi**, **clock, async** (set/reset).

Setup and Hold

In general, setup and hold measurements involve sweeping the data-pin transition with respect to the clock-pin transition. Sometimes the data sweeps toward the clock causing the delay or slew to degrade, and sometimes the clock transitions first and the data sweeps backwards toward the clock until a glitch is detected on the output. The failure criteria can be a degradation in delay greater than **constraint_delay_degrade** (10%), a degradation in slew greater than **constraint_slew_degrade** (50%), or a glitch on the output greater than **constraint_glitch_peak** (10% of Vdd). By default, flip-flops use delay degradation for both setup and hold, and latches use delay degradation for setup and glitch peak for hold. Flip-flops can use glitch peak for hold by setting parameter **constraint_glitch_hold**.

In some cases, the preferred metric cannot be used to measure a constraint. In such a case the required metric is used instead. For example, a setup for an enable pin on a flip-flop going inactive can only be measured using the glitch peak metric rather than the normal delay degradation.

There are several algorithms available for measuring Setup and Hold time built into Liberate including: Pass/Fail Bisection, Delay based Bisection and combinational. Liberate will use the following flow to measure constraints:

Pass/Fail bisection: This method requires a passing and a failing logical simulation (switching and non-switching output) to bound the constraint measurement. If this method cannot

produce any vector for the constraint arc that passes the constraint check, and if **constraint_combinational**=0, then output 0 values for the constraint arc.

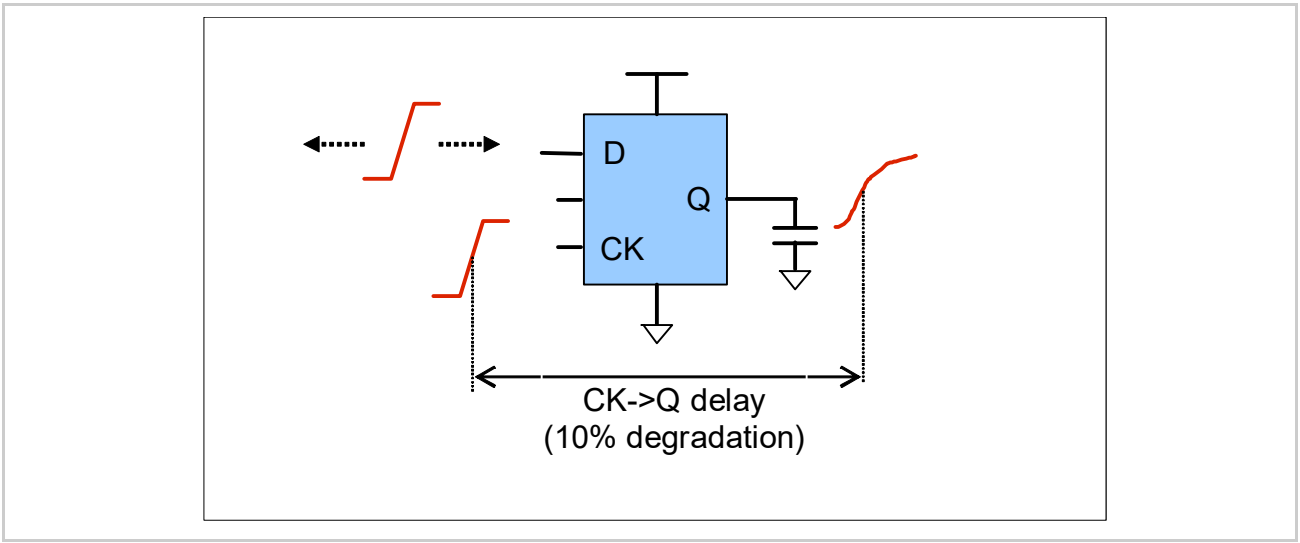
If **constraint_combinational**=1 and the pass/fail bisection fails, then use a delay based linear search for constraint characterization. In this method, a linear search is performed using a step size specified by the control parameter **constraint_combinational_step_size** where a pass and a fail are defined strictly by the delay degradation (without regard for logical pass/fail). If this method fails to characterize the constraint, then Liberate will output 0 values for the constraint arc.

If **constraint_combinational**=2 and the pass/fail bisection fails, then use a formula based method. This algorithm uses a formula that is based on the delay difference of the pin and related pin transition values. The formulae used are:

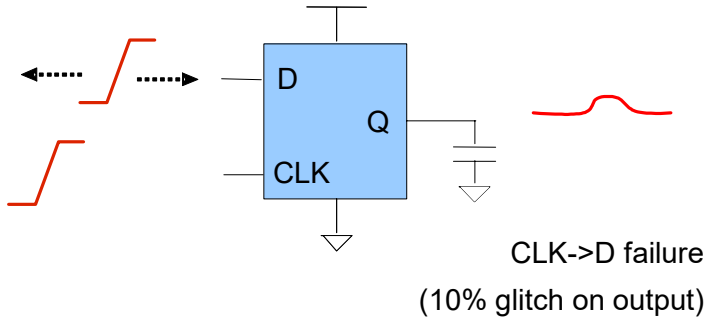
setup_rising rise_constraint value =
related_pin_transition - constrained_pin_transition + ${constraint_margin}
if < 0 then output "0" into the library
hold_falling fall_constraint value =
constrained_pin_transition - related_pin_transition + ${constraint_margin}
if < 0 then output "0" into the library

Note: constraint_margin defaults to 2ps

By default, the smallest output capacitance is selected as a load. This can be changed by using the constraint_output_load parameter to load the output pin to ensure conservative setup and hold values. If a cell contains multiple output pins, such as Q and QN, then each output pin will be loaded with the same capacitance value.

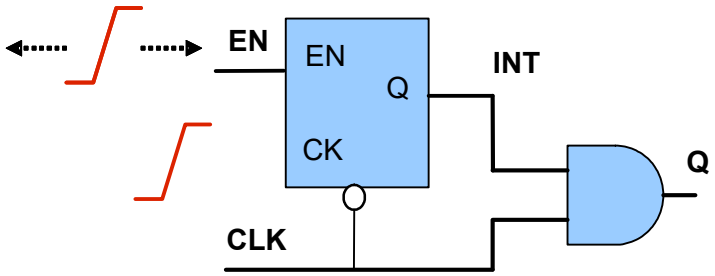


Setup and Hold Calculation



Hold Calculation for Latches

A smart binary search is used to find setup and hold. The search range is automatically determined by Liberate, based on proprietary circuit analysis techniques. Which output pin to monitor is set by the **constraint_output_pin** parameter. It can be an external pin or an internal node.



Typical Clock Gater Circuit

Clock-gating circuits such as the example shown in the figure above require special setup and hold measurement criteria.

For Setup-1, EN rising is swept toward a rising CLK. Then EN->INT and CLK->Q degrade, but CLK->Q is the critical path, so choose CLK->Q degrade as the failure criteria.

For Hold-1, EN Falling sweeps backwards toward CLK rising. Then INT glitches and eventually falls, and then Q delay degrades. But since INT glitches first, use that as the failure criteria.

For Setup-0, EN falling is swept toward a rising CLK. Then EN->INT degrades, but depending on the internal race between INT falling and CLK rising, output Q may glitch, so monitor both EN->INT delay degradation and Q glitch.

For Hold-0, EN rising is swept backwards to a rising CLK. Then INT will at first glitch and eventually rises, and then Q glitches. But since INT glitches first, use that as the failure criteria.

These special checks for clock-gater circuits can be disabled using **constraint_clock_gater** set to a "0".

Dependent Constraint Characterization

In normal circumstances, Liberate characterizes the setup time assuming that the hold time is infinite, as shown in *Diagram 1* below. Similarly, Liberate characterizes the hold time assuming that the setup time is infinite, as shown in *Diagram 2* below. These assumptions are based on the understanding that the data has a single transition relative to the clock. However, in a real system, the data can actually be a pulse where the data transitions both before and after the clock.

Diagram 1: Setup time (infinite hold)

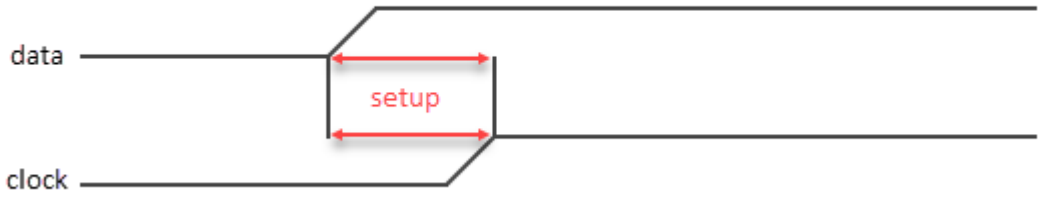
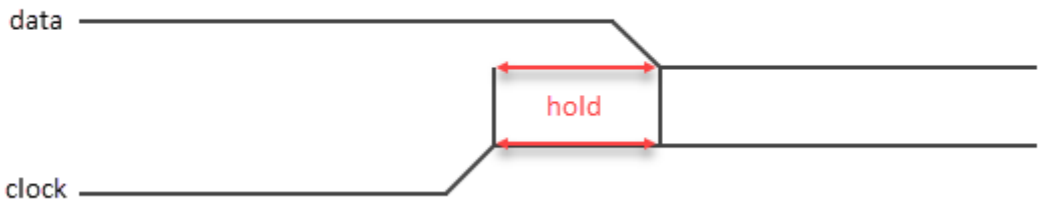


Diagram 2: Hold time (infinite setup)



When setup (hold) is measured in the presence of the hold (setup) time, that is, when the input data pin is driven using a pulse, the measured setup (hold) time increases significantly.

This is known as dependent setup (hold) where the setup (hold) time is dependent on a non-zero hold (setup) time as shown in *Diagram 3* and *Diagram 4* below.

Diagram 3: Dependent setup

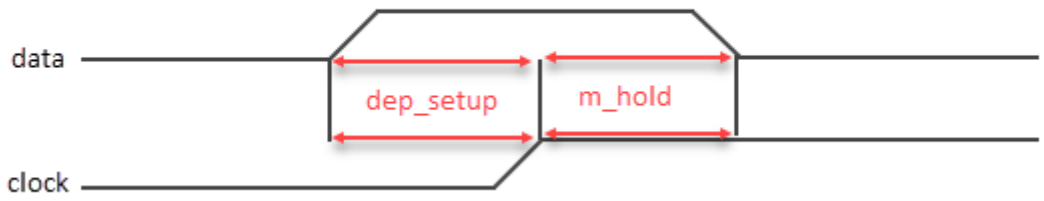
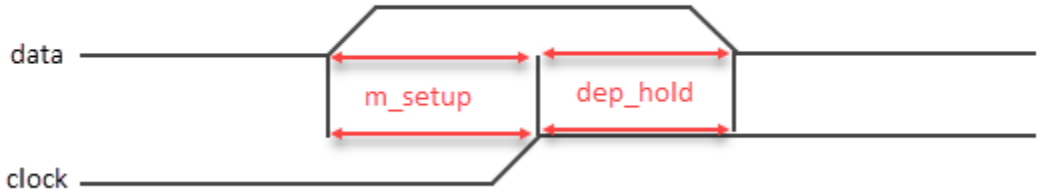


Diagram 4: Dependent hold



It is recognized that if the dependent setup (hold) time is measured while using the exact hold (setup) time reported by Liberate as in diagrams 1 and 2 above, the dependent setup and hold time will be extremely pessimistic to the point of being unrealistic. To remediate this extreme pessimism, Liberate provides the means to increase the pulse width of the input data by adding a margin to the pulse. The setup (hold) can be *margined* (that is, increased) per the following formula:

Margined Setup

```
m_setup =
    (setup +
    constraint_dependent_setuphold_margin +
    (constraint_dependent_setuphold_margin_ratio ( pin_slew + related_pin_slew )))
```

Margined Hold

```
m_hold =
    (hold +
    constraint_dependent_setuphold_margin +
    (constraint_dependent_setuphold_margin_ratio ( pin_slew + related_pin_slew )))
```

When dependent setup/hold is enabled, the data input is a pulse. The input pulse can narrow until, in some cases, it is reduced to a triangular waveform. The peak of this input triangular

waveform can drop away from the supply voltage. Use the constraint_dependent_setuphold_input_threshold parameter to specify how far the input pulse is permitted to drop away from the supply voltage.

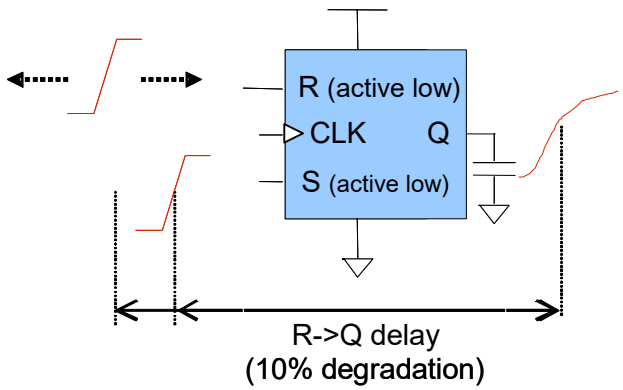
Recovery and Removal

Recovery and removal measurements are analogous to Setup and Hold, with the data pin replaced by an asynchronous set or reset pin. Recovery and removal between two asynchronous set and reset pins are also captured in a similar fashion. These models appear in Liberty format as timing_type recovery and removal. Control of the tolerances used in acquiring removal constraints can be controlled separately from hold tolerances by using the parameter **removal_glitch_peak.**

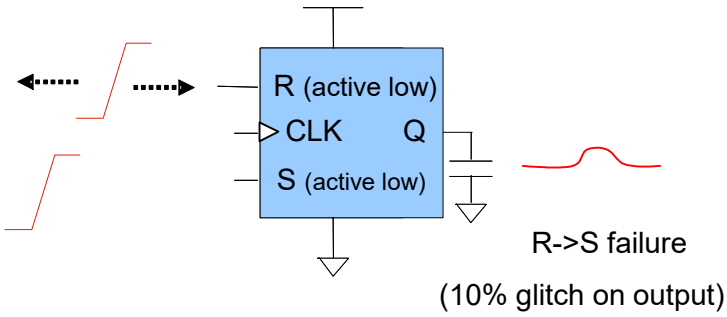
Non-sequential Setup and Hold

Non-sequential Setup and Hold measurements are analogous to setup and hold, but involve two asynchronous set or reset pins. These models appear in Liberty format libraries as timing_type non_sequential_setup and non_nonsequential_hold. To change the timing_type to recovery and removal, set the parameter **nonseq_as_recrem** to **1**. All Non-sequential Setup and Hold measurements are done using the de-assertion edges of both asynchronous pins.

The figures below illustrate how constraints are measured for two asynchronous pins. In this illustration, the Reset (R) pin is dominant, and the constraint is S->R. If a constraint is measured for a pin that is not dominant, such as R->S, it may not be possible to measure a change in the output pin. In such a case, Liberate forces the measurement of S->Q by finding an internal node that toggle as S toggles.



Non-sequential Setup acquisition



Non-sequential Hold acquisition

Min Pulse Width

Minimum pulse width is measured for all clock and asynchronous set and reset pins. The slew to use for the minimum pulse-width characterization can be defined using the mpw_slew parameter. This slew is used for both rising and falling transition of the **clock** or **async** signal. The mpw_slew_clock_factor can be used to change the slew for clocks to a ratio of the `mpw_slew`. The ratio of the fall to the rise slew can be changed using the mpw_skew_factor parameter. Setting this to less than 1.0 (default) will decrease the fall slew and increase the rise slew.

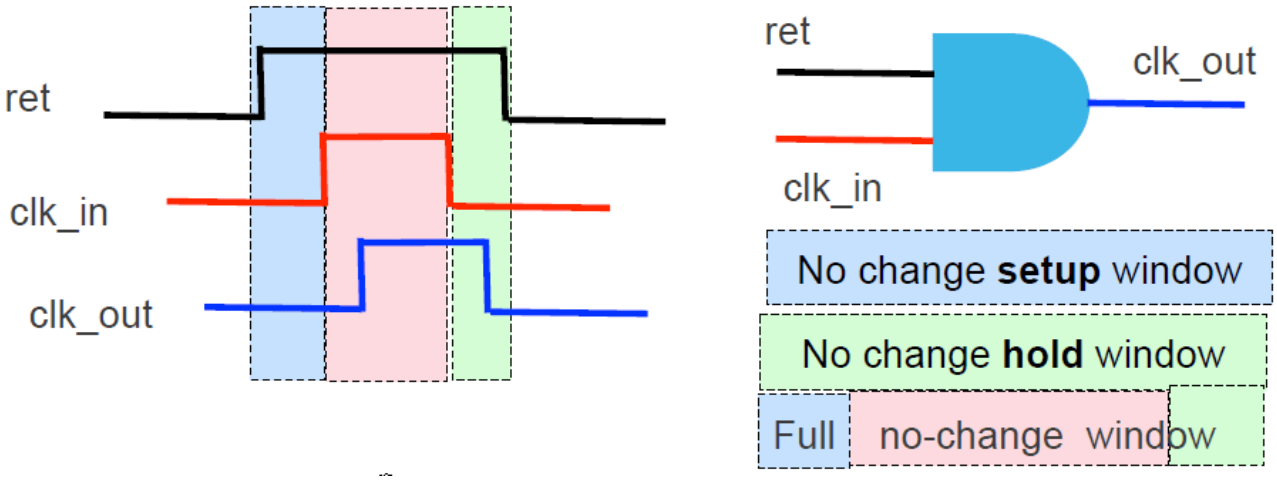
The minimum output load (taken from all the delay arcs related to this pin) is used as the load for calculating the minimum pulse-width. The minimum pulse- width is determined by a binary search, wherein the pulse-width will shrink until an appropriate failure criterion is met. There are 2 different cases. In one case, the output will normally switch once and the failure criterion in this case is met when the output fails to switch. In another case, the output will pulse like the input and the failure criterion is met when the output peak voltage fails to meet the glitch-peak as set by the mpw_glitch_peak control parameter. In both cases, the final output voltage is checked that it is at the correct voltage. The `constraint_output_pin` parameter specifies which output pin or internal node to monitor. The minimum pulse height permitted when the clock pulse becomes triangular can be set using mpw_input_threshold. If the input waveform peak drops below this threshold before the mpw_glitch_peak failure threshold is reached, then the characterization will end. The reported `min_pulse_width` value will be determined from the final input waveform.

By default, Liberate generates state-dependent *min_pulse_width* attributes. Set the mpw_table parameter to generate tables of min pulse width values based on the input slew of the clock. Note using tables will increase run-time.

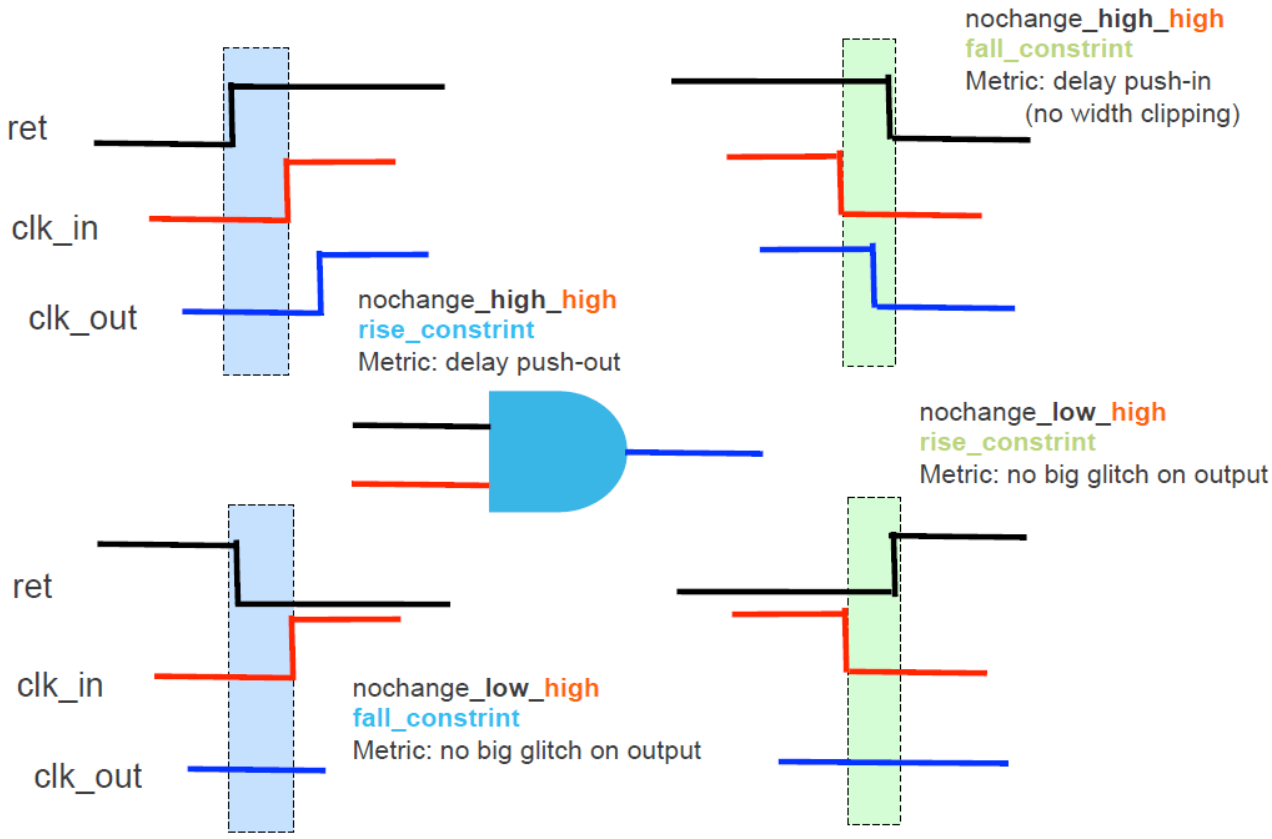
The mpw constraint will be filtered by direction for clock gater circuits, depending on the setting of the *clock_gating_integrated_cell* attributes.

No Change Arcs

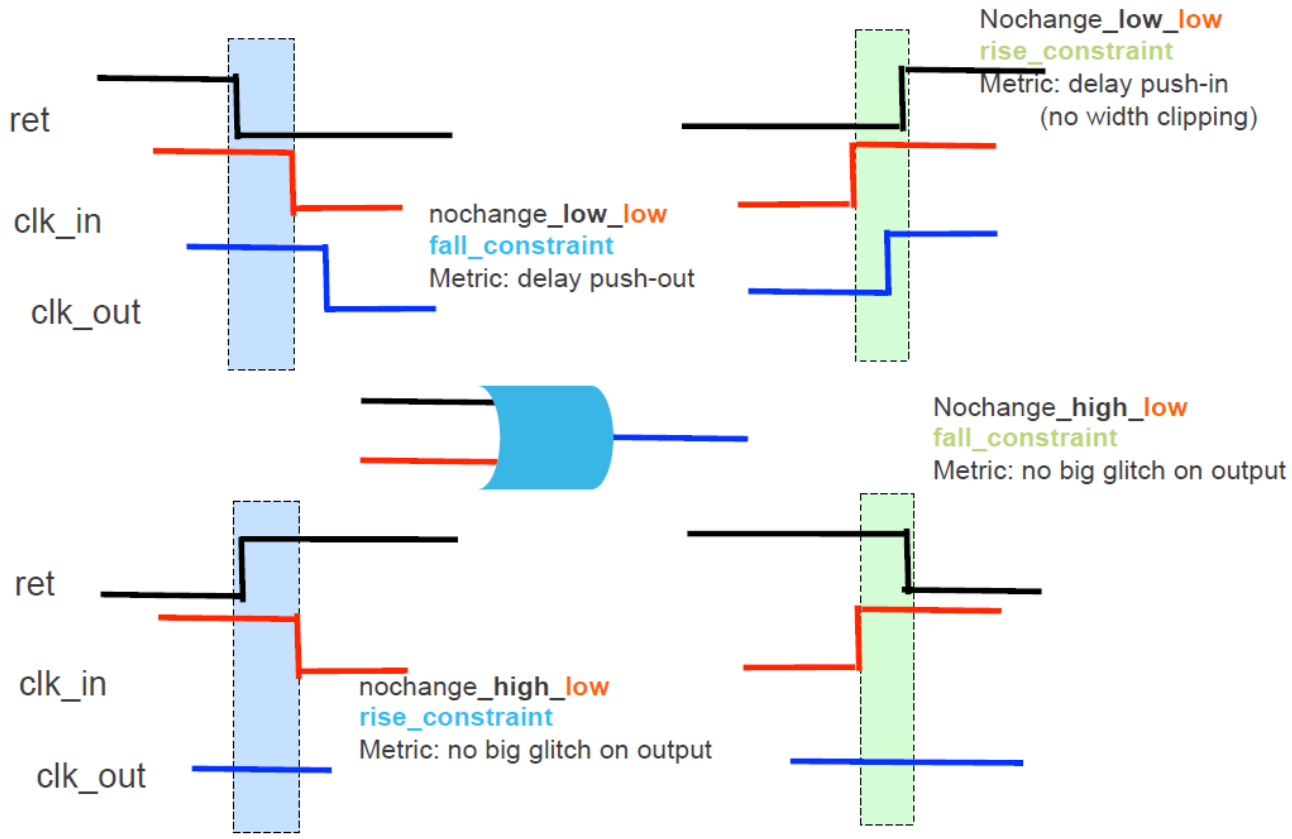
No change timing constraint checks a constrained signal against a level-sensitive related signal. The constrained signal must remain stable during an established setup period, for the width of related pulse, and during an established hold period. The following figure shows an example of no change timing window.



No Change Constraint: Types (AND - gater cell)



No Change Constraint: Types (OR - gater cell)



You do not need to specify `-metric` on `define_search` explicitly. It is automatically identified by the tool. The following table mentions how to choose `-type` for no change arc and how `-metric` is used in charcaterization.

Table 7-1 No Change Arcs

Gater logic	Table - How to chose nochange specific -type on define_arc & metric used for char			
AND / NAND	**nochange_high_high**	**nochange_low_high**	**nochange_high_low**	**nochange_low_low**
en & clk / !(en & clk)	delay	glitch	x	x
!en & clk / !(!en & clk)	glitch	delay	x	x
en&!clk / !(en&!clk)	x	x	delay	glitch
!en&!clk / !(!en&!clk)	x	x	glitch	delay
OR / NOR	**nochange_high_high**	**nochange_low_high**	**nochange_high_low**	**nochange_low_low**
en + clk / !(en + clk)	x	x	delay	glitch
!en + clk / !(!en + clk)	x	x	glitch	delay
en + !clk / !(en + !clk)	delay	glitch	x	x
!en + !clk / !(!en + !clk)	glitch	delay	x	x

No Change Constraint: Char Methodology

- The characterization methodology remains the same for setup and hold characterization.
- The characterization is based on a single edge of data or clock and there is no pulse passed in simulation.
- The clock-to-out delay push-in method is the same as delay push-out method except that the delay change expected is in the other direction (that is lesser than the reference delay).

No Change Constraint: Control/Setup

- Set the following parameter before `char_library`.

  ```
  set_var nochange_mode 1
  ```

  If this parameter is set to `0` and you have `nochange* define_arc` specified then you can control the modeled value.

- Relevant nochange arcs should be added in template file:

  ```
  define_arc -type nochange_high_high -related_pin CK -pin EN $cell
  define_arc -type nochange_low_high -related_pin CK -pin EN $cell
  ```

- All nochange types supported are:
  - nochange_high_high , nochange_low_high
  - nochange_high_low , nochange_low_low
- How to chose one of the above two sets for `define_arc`

See Table 7-1, “No Change Arcs,” on page 931 for using the correct combination based on your cell function and logic.

No Change Constraint: Modeling

```
pin (EN) {
direction : input; timing () { related_pin : "CK";
timing_type : nochange_high_high;
rise_constraint (constraint_template_3x3) {
}
fall_constraint (constraint_template_3x3) {
}

}
timing () {
related_pin : "CK";
timing_type : nochange_low_high;
rise_constraint (constraint_template_3x3) {
}
fall_constraint (constraint_template_3x3) {
}

}
documented for CCR No.
```

Power Models

Power is modeled in a Liberty file in separate sections. Power is divided into leakage, hidden (internal_power under the input pin), and power (internal_power for switching outputs that is found under an output or bidi pin). The following sections will discuss how Liberate models these power types and the parameters available to control the modeling.

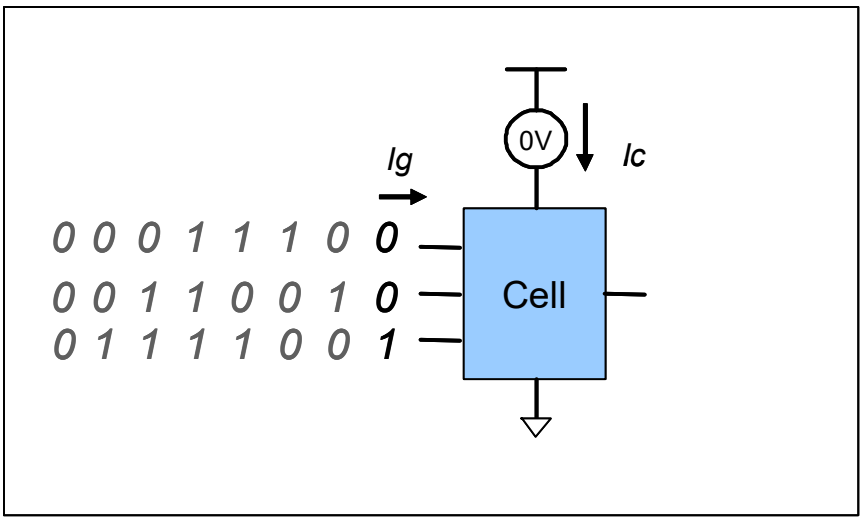
Liberate can completely ignore the power contribution for a specific supply pin. To ignore the power for a specific supply pin, add the **ignore_power** option to the set_gnd commands.

Leakage Power

All possible input logic combinations are evaluated for state-dependent leakage characterization. Both channel leakage, **Ic,** (the dominant leakage component in today's technology) as well as gate leakage, **Ig,** (projected to grow significantly in future technologies)

are characterized. The reported leakage power is the sum of the quiescent current from the supply and the current from logic high inputs, multiplied by the supply voltage. Use the **define_leakage** command to specify the desired leakage states to characterize.

The correct internal voltage is determined by the circuit simulator with **.nodeset** applied to each internal and output node. The simulator performs the DC solution of the circuit, after which the currents **Ic** and **Ig** are measured. The circuit used to measure leakage is shown below:



Circuit Used for Leakage Power Measurement

The `leakage_mode` parameter can be used to control the voltage applied when computing the leakage. By default, the measured dc current will be multiplied by the supply voltage. This will result in a leakage value of 0 for all gnd supplies operating at 0 volts. For other supported modes, see the leakage_mode parameter.

The power_subtract_leakage parameter can be used to enable the subtraction of the leakage power from the internal and hidden power. This parameter defaults to 1 which will trigger the subtraction of the average of the pre and post transition leakage power when voltage_map is not enabled. See the `power_subtract_leakage` parameter for the other modes that are available.

The pin_based_leakage parameter can be used to output state-dependent leakage without the related power pin. This can help reduce the overall size of the output library. The toggle_leakage_state parameter tells Liberate to toggle all clock pins once before measuring leakage. The result is that the internal states will be initialized and the *when* in the leakage group will not include the output pin. The power_combinational_include_output parameter

can be used to directly control if the when in the output leakage group for combinatorial cells will include the output pin. The power_sequential_include_complementary_output parameter can be used to request that the when in the output leakage group for sequential cells will include complementary output pins.

Liberate will include the input pin DC leakage (Lpin_dc) in the value in the leakage groups. To not include the Lpin_dc in the leakage, set the leakage_add_input_pin parameter to `0`.

Liberate will limit the number of leakage states to a maximum of 256 states (vectors). This limit can be changed by setting the max_leakage_vector parameter to the desired maximum number of leakage states to characterize. If the default_leakage command is specified, then the *Inside View* algorithm of Liberate will be disabled for leakage characterization, and only the user specified leakage *when* states will be characterized.

The conditional_leakage parameter can be used to disable the output of state-dependent leakage groups.

The keep_dcap_leakage parameter can be used to request the characterization and subsequent reporting of leakage for decap cells.

Switching and Hidden Power

Switching power (see `define_arc -type power`) measures the energy dissipated by the cell when one or more inputs switch which causes one or more outputs to switch. It includes short-circuit power plus internal switching power that is dissipated during the charging and discharging of internal capacitive loads. The energy dissipated via the non-switching input pins is also measured as it contributes to the total dissipated energy inside the cell.

Hidden power (see `define_arc -type hidden`) measures the energy dissipated inside the cell when one or more inputs transition and do NOT result in any output transition. Hidden power is reported in the Liberty file as an `internal_power` group, usually state dependent, in an input pin group. If more than one input is switching, as might be the case when the cell has dual inputs, the hidden power reported will be divided by the number of switching inputs. The parameter hidden_power can be used to enable or disable hidden power characterization for all combinational cells. Sequential cell inputs will always have `hidden_power`. By default, hidden power will be calculated for all input pins including combinational gate inputs and will be reported only with full state dependency. No default hidden power group will be output. The user can disable state dependency for hidden power with the parameter conditional_hidden_power. When this parameter is set to `0`, no state-dependent hidden power will be reported. Instead, only a default (state independent) hidden power will be reported. It is sometimes desirable to subtract hidden power from switching power. To enable this, use the subtract_hidden_power parameter. When this feature is

enabled, and hidden power has been characterized, then Liberate will choose the first matching hidden power value to subtract from the switching power in the output pin group.

Current is continuously monitored through all Vdd (see **set_vdd**) and Gnd (see **set_gnd**) supplies from the beginning of the input transition to the end of all the internal and output pin transitions (0.5% of supply for falling transition and 99.5% of supply for rising transition). For both rising and falling transitions, the energy dissipated during the charging or discharging of the output load capacitor is subtracted from the total energy calculation (see power_subtract_output_load). For cells with multiple output pins, the energy dissipated by other switching outputs is also accounted for. The parameter pin_based_power can be set to `0` to only include power dissipated via the positive power supplies (see **set_vdd**), excluding the power in the non-switching input pins. This can be useful for correlating power results between libraries from other characterization tools. Alternatively, when this parameter is set to a `2`, Liberate will monitor the current (including Miller capacitance) to each cell input. If the input is tied to a positive/negative voltage, that current is added to the vdd/gnd power. This non-switching input pin current is not accounted for in any other power measurement. This is because when the driving cell is characterized, the receiving cell is not included, and even if it was, it may not be switching in the correct way to trigger the Miller capacitance currents.

Liberate will normally output power that is not related to a specific power pin. Set the parameter voltage_map (default=0) and the parameter pin_based_power (default=1) to request that power be reported by power pin. With these parameters set, the output library will have voltage_map groups, pg_pin groups, and the power will be reported with the related_pg_pin attribute.

Liberate will check that the sum of the rise_power and the fall_power will not be negative. If the sum is negative, then Liberate will adjust the more negative value such that the sum becomes zero. The parameter reset_negative_power can be set to `0` to disable this check.

Power Subtraction

Non-Linear Power Models (NLPM) are often generated with the understanding that downstream power tools will sum the data from multiple power groups when reporting power for a specific power event. To prevent power from being over-reported, Liberate can subtract leakage power and/or hidden power. See power_subtract_leakage and subtract_hidden_power for more details.

When the *Inside View* algorithm of Liberate generates the characterization plan or if all arcs are manually specified (see define_arc) with full state dependency, then power subtraction can be a straight-forward process to characterize, model and validate:

```
Switching Power (internal_power under an output pin) = Simulation value - hidden power - leakage power
```

Hidden Power (internal_power under an input pin) = Simulation value - leakage power

However, if the characterization plan does not include complete state coverage then selecting the correct hidden or leakage power can be problematic. The most common cause is a verbose template that does not have full state coverage. This does not mean that the power in the library is incorrect, but it may mean that it is more difficult to validate the reported power values.

If an exact leakage state, including the states of output pins, is not included in the characterization plan, then it will not be included in the output library. If leakage power subtraction is enabled, then Liberate will select either a leakage state that has a full or partial overlap or a default leakage power. For example, if the define_leakage commands only include WHEN states for input pins, then any combination of states on the output pins would still allow for a match.

When power tools add leakage power, they will select the best match for the actual state that is represented in the model. On large cells, there can be a difference of 60% leakage power from state to state, so it is important to have Liberate subtract the same leakage vector that the power tool will add back in, even if this means that the power number represented in the model does not necessarily match the SPICE simulations.

The problem is more complicated when subtracting hidden power from switching power arcs. It can cost a prohibitive amount of runtime to characterize and model all possible combinations of input and output pin states for hidden power in order to be able to validate any given state of switching power. To reduce this burden and maintain accuracy, some users will write specific define_arc statements that align vectors used for hidden power and switching power. If the vectors are not aligned, the power values will be more difficult to validate.

Power Validation

Validating power calculations is one of the challenging aspects of library qualification, especially since relevant power data may not be contained in a single SPICE simulation. To reduce the burden, Liberate can print power equations and associated values for validation purposes. See power_info for details on how to enable this feature.

Common Usage Modes for Power and Leakage

No PG-pin syntax

1/2 of Vdd, Gnd, input energies, subtract 1/2 of output load energy

```
set_var voltage_map0      # (default)
```

```
set_var pin_based_power2
set_var power_subtract_leakage1      # (default)
set_var power_subtract_output_loadall    # (default)
set_var leakage_add_input_pin1      # (default)
set_var leakage_mode1      # (default)
```

PG-pin syntax

1/2 of Vdd, Gnd, input energies, subtract 1/2 of output load energy

```
set_var voltage_map1
set_var pin_based_power2
set_var power_subtract_leakage1       # (default)
set_var power_subtract_output_loadall     # (default)
set_var leakage_add_input_pin1       # (default)
set_var leakage_mode1       # (default)
```

No PG-pin syntax

Vdd energy, subtract output load energy on rise

```
set_var voltage_map0       # (default)
set_var pin_based_power0
set_var power_subtract_leakage1       # (default)
set_var power_subtract_output_loadall     # (default)
set_var leakage_add_input_pin1       # (default)
set_var leakage_mode1       # (default)
PG-pin syntax
```

Vdd energy, subtract output load energy on rise

```
set_var voltage_map1
set_var pin_based_power0
set_var power_subtract_leakage0       # (default)
set_var power_subtract_output_loadall     # (default)
set_var leakage_add_input_pin1       # (default)
set_var leakage_mode1       # (default)
```

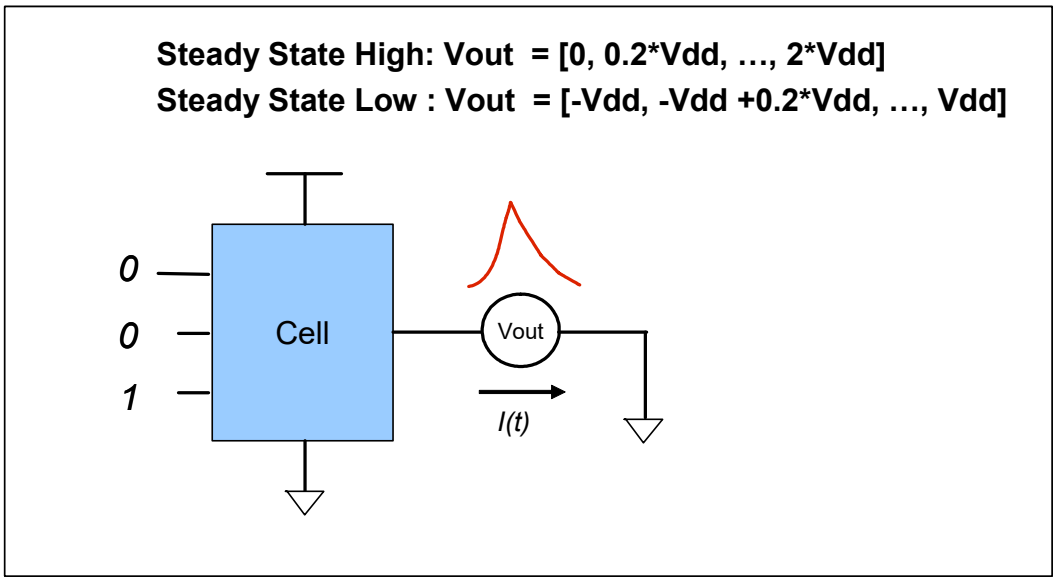
Signal Integrity Models

Liberate supports both Liberty SI models and composite current-source noise models (CCSN). The Liberty SI model contains constructs for steady state current, DC noise margins, hyperbolic noise curves and noise immunity tables. To characterize Liberty SI models, use the **si** option to **char_library**.

CCSN models are characterized for each channel-connected collection of transistors (stage) that connects to each input, inout or output pin. For timing arcs that have two or less stages between the input and the output pin, the CCSN data is stored on each timing arc. For timing arcs with more than two stages between input and output, such as flip-flops, the CCSN data is stored on the appropriate input, output or inout pin. There are four components to the CCSN model: DC current, output voltage, Miller capacitance and propagated noise. To characterize CCSN models use the **ccsn** option to **char_library**.

Steady State Current

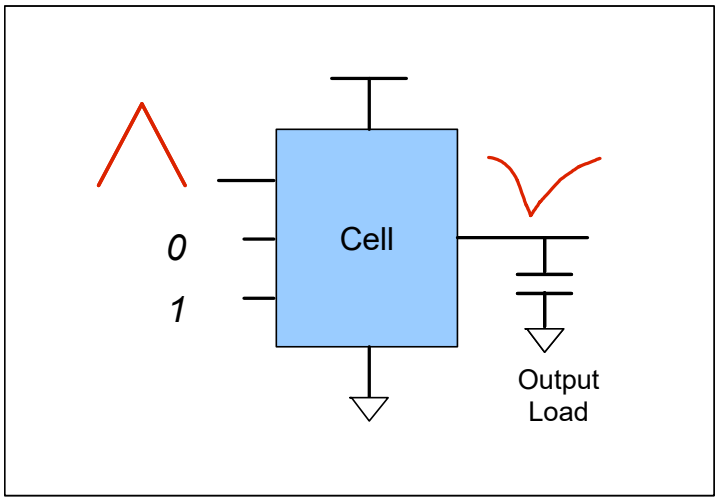
Steady state current is characterized for each cell output/inout pin by putting the output into steady state, connecting a voltage source to that pin and varying the voltage level of the voltage source from 0 volts to 2*Vdd, for steady-state high, and from –Vdd to Vdd, for steady-state low. The current through the voltage source is captured in the steady-state current table. This is performed for each distinct logic state where the output goes low or high. The intermediate voltage values to be tabulated can be specified by defining a template of type `si_iv_curve`.



Circuit used for Steady State Current Calculation

Noise Immunity Curves

Noise immunity is characterized for each cell input/inout pin by injecting input glitches of various widths and heights at that pin for a set of output loads. For each width/load combination, a search is performed to determine the input glitch- height that causes a noise failure (where a failure is defined by a propagated noise pulse whose height exceeds the **immunity_glitch_peak**, default 5% of Vdd). The noise-immunity curve is a table of failure heights for the widths and loads defined by the given template of type **si_immunity**. A unique noise-immunity curve is generated for each valid logic state.



Circuit used for Noise Immunity

The input noise-glitch used is assumed to be an isosceles triangle, unless skewed using the **immunity_noise_skew_ratio** parameter. Characterizing for noise-immunity is enabled by the **si** option to **char_library** and by defining a **si_immunity** template for that cell.

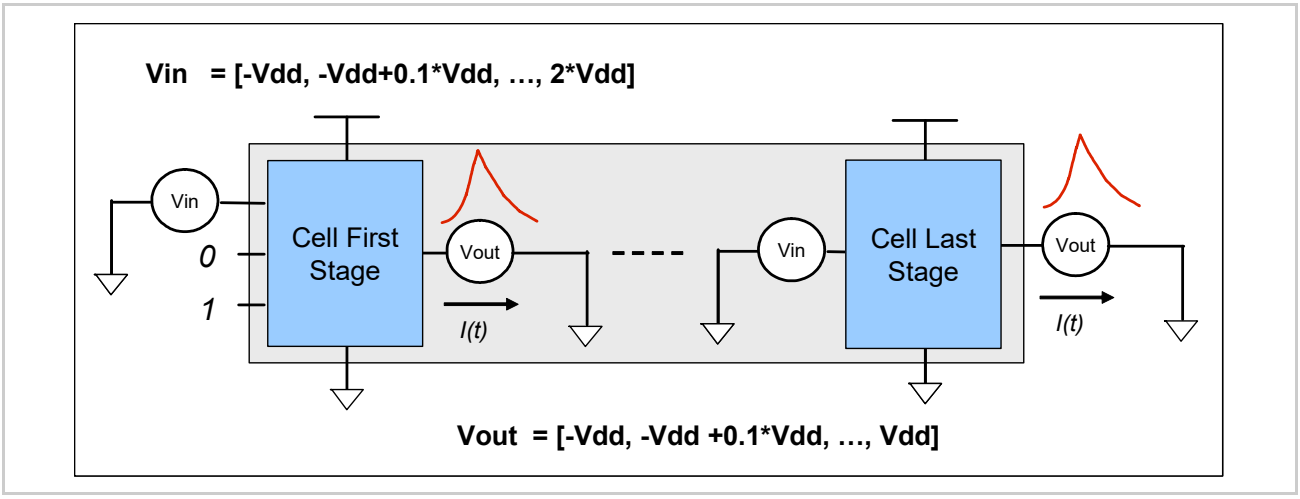
Hyperbolic Input Noise, DC Noise Margin

In addition to the noise-immunity curve, hyperbolic input noise (low and high) and DC noise-margins can be generated in the output library by setting the parameter **input_noise**. These values are extracted from the data captured by calculating the noise immunity curve. The hyperbolic area, height and width are determined by fitting the noise-immunity curve for the minimum load to a hyperbolic function using a least-squared fit. The DC noise margin is the height of the hyperbolic curve.

Composite Current Source Noise (CCSN) Models

CCSN DC Current

The CCSN DC current model is calculated for both the input (first) and output (last) stages of a cell. For each stage, under every logic condition, a simulation is performed where the input to the stage (input/inout pin or internal node) and the output of the stage (output/inout pin or internal node) are both connected to a voltage source. Each voltage (input and output) is varied from –Vdd to 2*Vdd and the DC current through the output voltage source is recorded. The voltage steps between –Vdd and 2*Vdd are determined by a pre-defined formula, such that 29 voltage points are sampled for each input and output voltage resulting in a 29X29 DC current table.

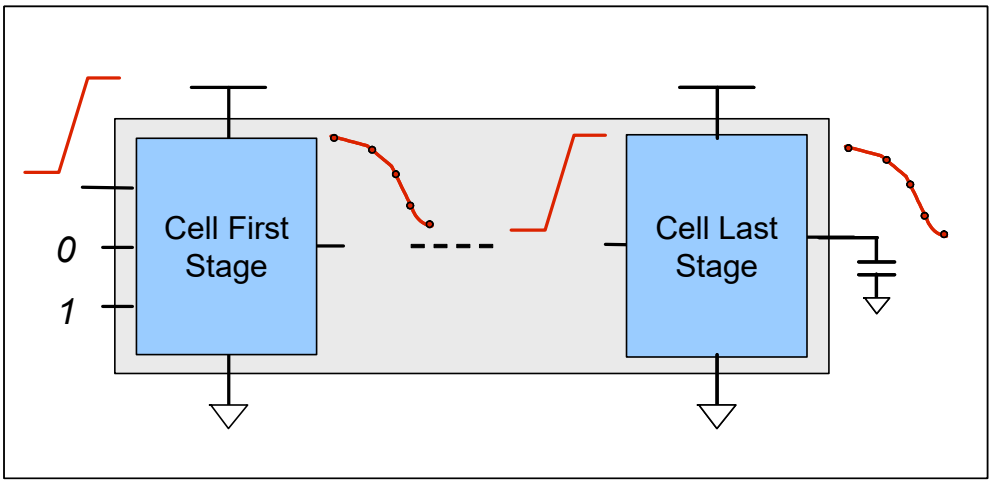


Circuit used for CCSN DC Current

CCSN Output Voltage

The CCSN output voltage model is calculated by simulating each stage using two linear slews from the input slew indexes, and either two loads from the output load indexes (if the stage output is a pin) or zero additional load capacitance (if the stage output is an internal node).

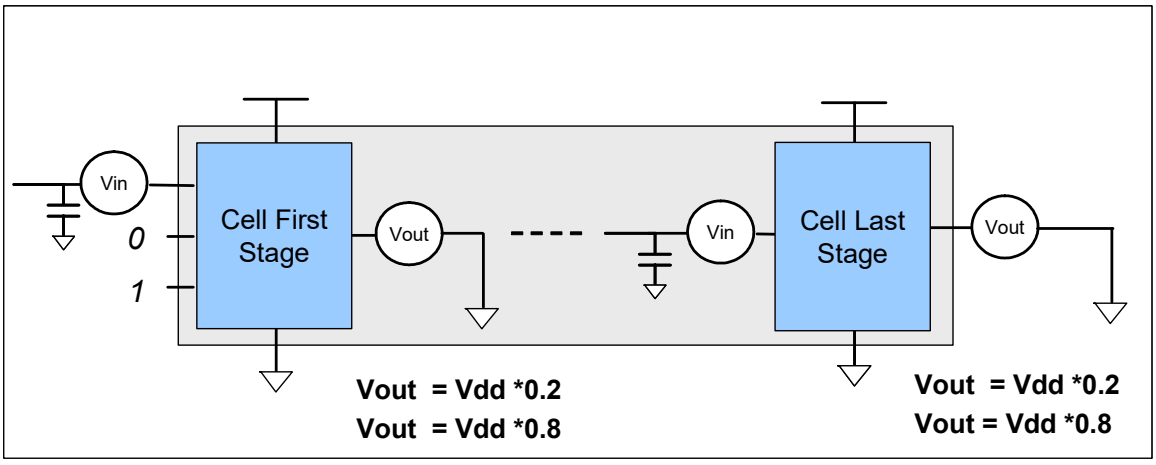
Consequently, each stage is simulated either two or four times. The output voltage at the stage output is measured at five time-points.



Circuit used for CCSN Output Voltage

CCSN Miller Capacitance

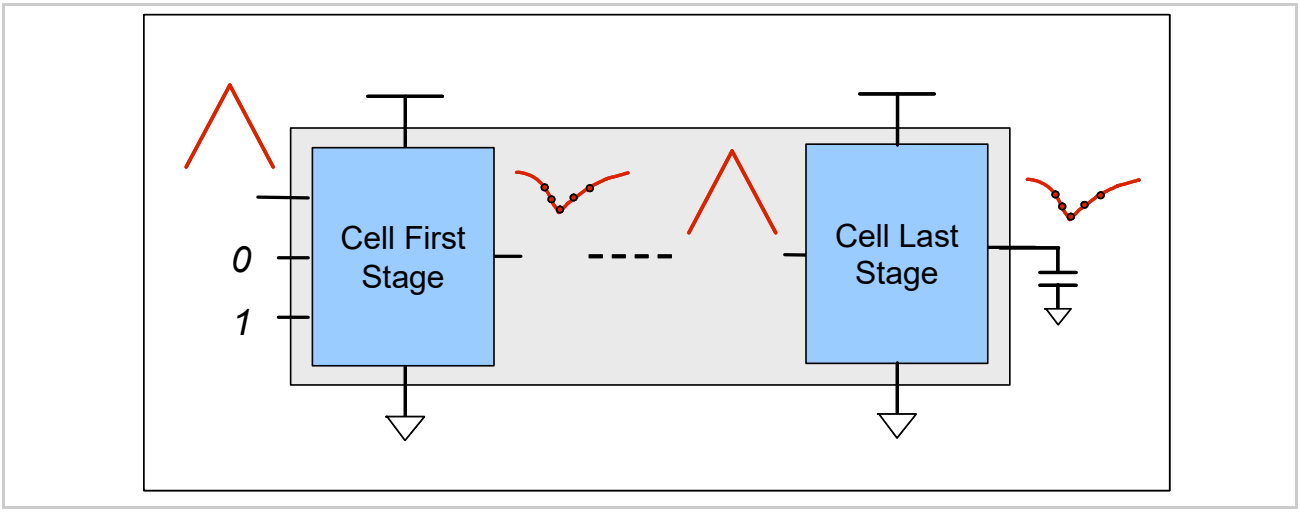
The CCSN Miller capacitance is calculated by placing a voltage source at the output of each stage and a capacitive load at the input pin of the stage. The Miller capacitance is measured by comparing the change in input voltage against a 20% change in output voltage for two different input loads.



Circuit used for Miller Capacitance

CCSN Propagated Noise

The CCSN propagated noise model is calculated by simulating input noise glitch triangles, of various widths and heights, over a small range of capacitive loads for each stage. If the output of a stage is a pin, two loads are used, otherwise a single zero load capacitance is used. Input glitches of the appropriate height and width are automatically determined, to ensure that there is a propagated noise pulse at the stage output. The resulting output glitch voltage is captured using 5 points - 50% of the leading edge, 80% of the leading edge, the glitch peak, 80% of the trailing edge, 50% of the trailing edge.



Circuit used for CCSN Propagated Noise

Electromigration Models

When high-density current passes through a thin metal wire, the high-energy electrons exert forces upon the atoms causing them to *migrate*. This electromigration (EM) can drastically reduce the lifetime of an electrical circuit, by causing increased resistivity or a break in the metal wire or by creating a short circuit between adjacent lines. Electromigration can be controlled by establishing an upper boundary or maximized for the output toggle rate. This is achieved by annotating cells with electromigration characterization tables that represents the net's maximum allowable toggle rates.

In Liberty libraries, the electromigration pin-level group attribute contains the `em_max_toggle_rate group`, which specifies the maximum toggle rate, and the `related_pin` and `related_bus_pins` attributes. The `related_pin` attribute associates the electromigration group with a specific input pin. The `related_bus_pins`

attribute associates the electromigration group with the input pin or pins of a specific bus group.

Electromigration Characterization Flow

The EM characterization flow in Liberate requires the use of Cadence Spectre APS/EMIR, an EM configuration file, and SPICE subcircuit descriptions in DSPF format in addition to the other commands normally found in a Liberate characterization setup.

For creation of an EM liberty model in Liberate:

1. Create a valid Liberate setup for all cells requiring EM models. The SPICE subcircuits loaded into `read_spice` are required to be in DSPF format. The setup must use Spectre-APS and have sufficient MMSIM licenses (compatible with Spectre EMIR) available.

   Note: The `Spectre_Char_Opt` license is not usable with EM characterization.

2. Provide the additional EM commands to the Liberate setup. At a minimum, this includes the maximum clock frequency (see em_clock_freq) and the QRC technology file (see em_tech_file).

3. Execute Liberate using the currently supported version of Spectre (see `${ALTOSHOME}/README`) or a newer version.

4. Characterize the library for EM by adding the `-em` option to the char_library command.

5. Generate a liberty library with EM data by adding the `-em` option to the write_library command.

6. If needed, merge the EM data into another library that has all of the data formats, including the more advanced formats such as CCS and ECSM.

Gathering Required Data for EM Characterization and Modeling

The enhanced EM capability in Liberate was developed to take advantage of the Spectre APS/EMIR capability. As such, Liberate requires that the data needed by Spectre APS/EMIR be provided in the Liberate Tcl file. The data needed consists of a DSPF format netlist file for each cell to be characterized, process models for the process/voltage/temperature (PVT) corner of interest, an EM data file or QRC tech file and a version of Spectre APS that supports EMIR (version MMSIM 13.1 ISR13 or later). In addition, the user must provide a maximum clock frequency for the library.

The interface between Liberate and Spectre APS/EMIR was designed such that the creation of the spice decks, the simulation, and parsing of the simulation output is fully automated.

Creating the Run Script

Obtain a fully validated Liberate characterization script for the library. To enable EM characterization, add the following two parameters to the script prior to the **char_library** command. They will define a location for the EM QRC technology file required by Spectre APS EMIR and a maximum clock frequency to use for the spice level simulation. The syntax and usage is as follows:

```
# EM related clock frequency in Hertz
set_var em_clock_freq  20e6
# Specify the full path and filename of QRC tech file
set_var em_tech_file /proj/work/tech.qrc
# Request Spectre to use APS
set_var extsim_cmd_option "+aps -mt +spice"
# Tell Liberate to use Spectre
char_library -extsim spectre
```

Note: There are other parameters that can be used to tune the EM characterization. For more information, see the parameters prefixed with `em_` in Chapter 5, "Liberate Parameters."

Creating a Liberty Library with EM Data

To create a library with EM data, add the option `-em` to the **write_library** command.

Example:

```
read_ldb MyEM.ldb
write_library -em -filename My.lib typical
```

Merging EM Data into an Existing Liberty Library

To merge a library with EM data into another library, use the **merge_library** command.

Example:

```
merge_library My.lib MyEM.lib
```

EM Characterization Example

The liberty syntax for EM data consists of input slew/output load related toggle rates for an output pin. They can either be a scalar value or a table of toggle rates based on a variety of

input slews. The syntax of the Electromigration data in the Liberty format is as given in the following:

Example:

```
pin (O) {
    electromigration () {
        related_pin : i0;
        em_max_toggle_rate (em_lut_template_7x7) {
            index_1 (“slew1 ... slewN”);
            index_2 (“load1 ... loadN”);
            values ( \
                togglerate1 ... \
                togglerateN ... )
        }
    }
    electromigration () {
        related_pin : i1;
        em_max_toggle_rate (em_lut_template_7x7) {
            index_1 (“slew1 … slewN”);
            index_2 (“load1 ..loadN”);
            values ( \
                togglerate1 ... \
                togglerateN ... )
        }
    }
}
```

Notice that each input pin has a separate electromigration group. For cells with multiple input pins, these groups can also be state dependent. Liberate will automatically create an electromigration group for each delay arc characterized for each cell.

Data Table Index Determination

The size of the characterized data tables are predetermined by the user from the define_index command. The define_template command must be specified and referenced in the define_cell command for each cell. All arcs for the cell will have the number of slew and load indexes as specified in the `define_template` command.

Liberate has 3 methods available for determining the actual data table index values, also known as `index_1` (slew index) and `index_2` (load index).

Method 1: Liberate uses the define_template index values.

The user specifies the exact indexes desired using the define_template and define_index commands. This method gives the best runtime since Liberate will not spend any CPU time determining the index values. The write_template command can be used to extract the exact index values from an existing library.

Method 2: Liberate computes the index values.

The user specifies the max_transition, min_transition and min_output_cap values. These values can be extracted from an existing library by write_template. Liberate determines the max_capacitance by simulating the arc for each cell from each input to each output (all "when" states are observed). The input pin will be driven using the max_transition. Liberate will adjust the output load until the max_transition is reached as measured at the measure_slew_* thresholds. Although it may not be required, the user should provide the min_transition (smallest input transition) and min_output_cap (smallest output load) values. If these not provided, then the min_transition will be set by Liberate to the smallest/fastest slew as estimated for all outputs in the library when unloaded. If not provided, the min_output_cap will default to an estimation of the smallest input cap (the transistor port cap + wire cap) for all input pins in the library. These min transition and load values must be provided when using a packet based flow because only one cell will be evaluated at a time resulting in tables for a cell that may not extend over the desired range for the library. Once the min/max transition and min load are determined, then the intermediate values in the table will be determined using a simple geometric series.

Method 3: The user wants to maintain a curve shape as exists in a current library, but wants to change the max_transition, min_transition and max_output_cap.

Liberate can read in the existing library (see read_library), and can output a template where the values in the define_template and define_index commands are a ratio of the index_n(max)-index_n(min) (see scale_tran_by_template and scale_load_by_template for details). The template file will have index_1/2 values that are ratios of the original index with the min value being 0 and the max value being 1. This flow can be useful if you want to increase the range of the data table. You can change the max value to a number greater than 1 or add additional values (increasing the number of values). By adding to or changing the values in the index, the data table can be made to extend beyond the design rules. For example, if the index_1 template has a max value of 1.5, then the data table will be characterized to a slew that is 1.5 times the max_transition value.

In methods 2 and 3 above, if the simulation method for determining the max_capacitance as described above should fail for some reason, then Liberate will set the max_capacitance using the method described in the documentation of the max_capacitance_auto_mode parameter.

Liberate will characterize the max_capacitance in methods 2 and 3 above using a single output transition. The specific arc used will be determined by a proprietary algorithm that looks at all input to output arcs and states. Set the parameter auto_index_distinct_risefall to have Liberate compute a separate `max_capacitance` for rise and fall transitions.

8

Improving Characterization Performance for Multi-Bit Cells

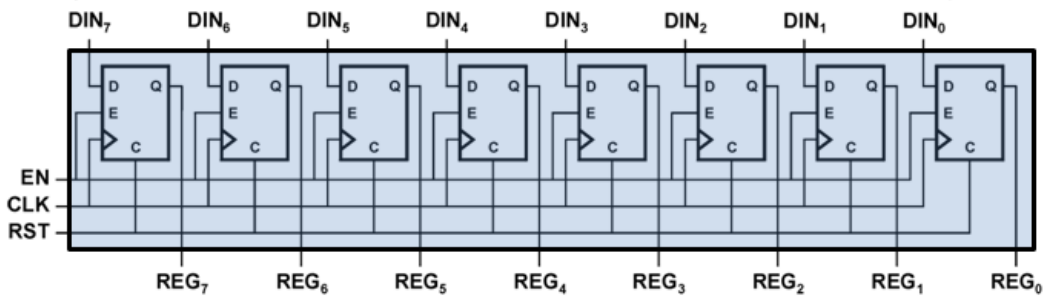
Overview

Using the patented *Inside View* algorithm, Liberate can automatically characterize multi-bit cells such as flip-flops, latches, or combinational logic. This automatic method can generate an exhaustive set of vectors or a set of vectors optimized for minimal run time with acceptable accuracy. An exhaustive set of vectors is desirable for 2-bit cells. For cells of 4-bit size or greater, exhaustive state conditions will result in long run time caused by simulations of all 2N combinations of input pins. Many state conditions are not needed to produce a complete library, and can be omitted to reduce the run time with little or no loss of accuracy. Liberate can automatically find the worst-case timing and power state conditions needed to ensure that a library is accurate.

Using the *Inside View* algorithm with user-defined vectors will direct the tool to only consider a limited set of state conditions. This method can be used to ensure worst-case timing and power coverage if the worst-case state conditions are known in advance.

Multi-Bit Cell Behavior

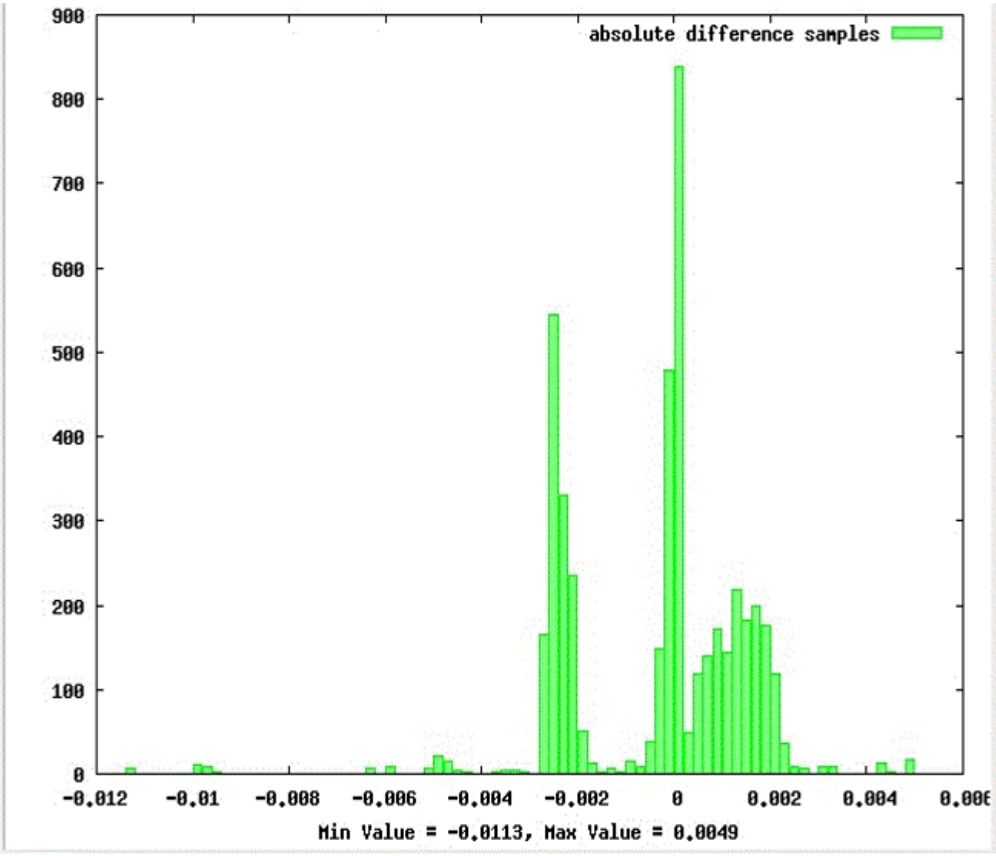
To understand the behavior of multi-bit cells, let us observe the following figure of a block-level schematic of an 8-bit DFF with shared *Reset* and *Enable*:



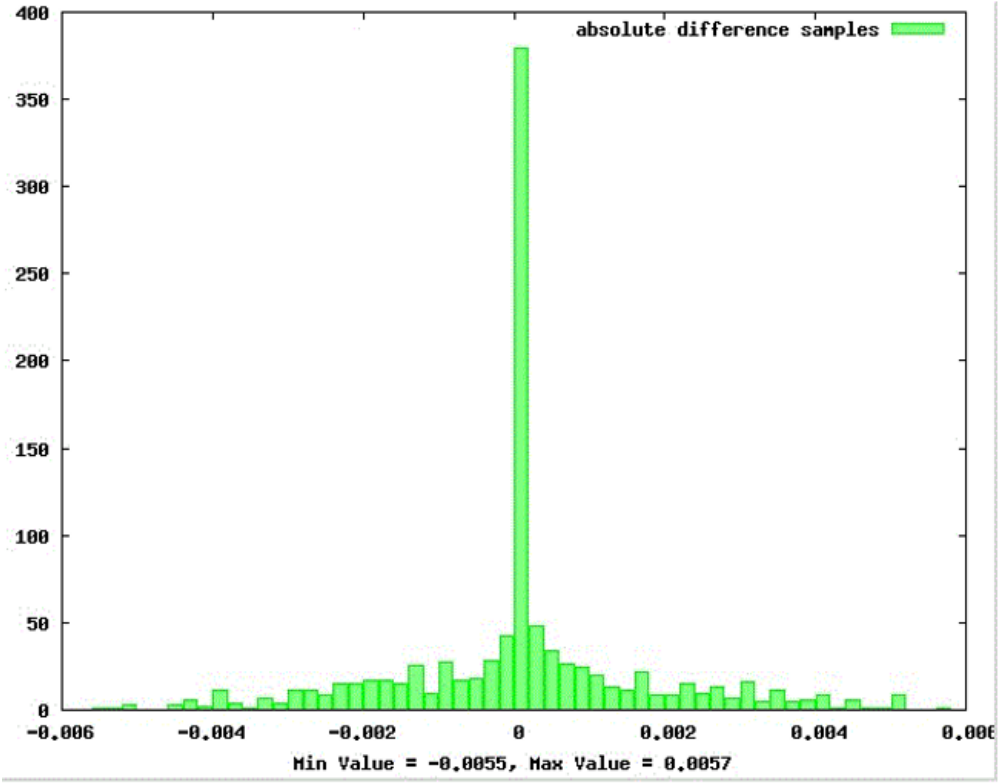
Such a cell would require 2^{10} state conditions to exhaustively cover all `CLK->REG` arcs. Input and output pin power could additionally depend upon internal states of the master and slave stages plus the state of all output pins. This will result in 2^{18} possible state conditions. However, only a small subset of these exhaustive timing and power states are needed to ensure accuracy.

Below are two histograms that show the bit-to-bit timing differences for a 4-bit Scan DFF cell. The bit-to-bit timing skew seen in such histograms will vary among designs, depending upon the level of care taken to ensure that signal propagation is symmetric among all bits. Asynchronous preset and clear paths often have larger skews. For this reason, bit-dependent timing is desirable.

In the following figure, `CLK->Q` timing skews are shown for a 4-bit Scan DFF among all bits for all the state conditions, slew, and load combinations. The X axis is in nanoseconds and Y axis is the population of delay differences.



In the following figure, `D->CLK` setup constraint timing bit-to-bit skews are shown for a 4-bit Scan DFF among all bits for all the state conditions, slew, and load combinations. The X axis is in nanoseconds and the Y axis is the population of differences.



Bundle Library Syntax for Multi-Bit Cells

The Liberty library bundle syntax allows multi-bit cells to be modeled with shared or bit-blasted models. The Liberate define_bundle_pins statement will cause the pins to be bundled in the library.

```
# characterize non-state-dependent timing and power

define_bundle_pins $cellname D {D3 D2 D1 D0}
define_bundle_pins $cellname Q {Q3 Q2 Q1 Q0}
char_library …
# Generate a library with bundle syntax for the D and Q pins
write_library -unique_pin_data
```

```
# bit-blasted models with unique timing on each bit
bundle (reg) {
    members (reg1, reg2);
    pin (reg1) {
    timing () {
        /* the related pin is a bundle */
        related_pin : "clk";
        cell_rise (delay_template) { values_1 } … };
    pin (reg2) {
    timing () {
        /* the related pin is a bundle */
        related_pin : "clk";
        timing_sense : non_unate; timing_type : edge;
        cell_rise (delay_template_2x2) { values_2 } … };
};
# timing models with all bits sharing common timing
bundle (reg) {
      members (reg1, reg2);
      timing () {
        /* the related pin is a bundle */
        related_pin : "clk";
        cell_rise (delay_template) { values_1 } … };
};
```

Bit-blasted models are desirable to account for the bit-to-bit timing skews found in most cells. If such accuracy is not important for a library, only a single bit can be simulated and the timing shared among all bits. For example, a library used only for synthesis at mature process nodes might be accurate enough using shared common timing. Bit blasted models are usually necessary for timing signoff accuracy, especially at advanced nodes.

Library Timing Structure

```
pin (Q0,Q1,…,QN) {
    related_pin : "CK";
    timing_type : rising_edge;
    when : "!R&SE";
    when : "!R&!SE";
    related_pin : "R";
    timing_type : clear;
    when : "CK";
    when : "!CK";  }
pin (CK) {
    timing_type : min_pulse_width;
    when : "!R&SE";
    timing_type : min_pulse_width;
    when : "!R&!SE"; }
pin (D0,D1,…,DN) {
    timing_type : hold_rising;
    when : "!R&!SE";
    timing_type : setup_rising;
    when : "!R&!SE"; }
```

```
pin (R) {
    timing_type : recovery_rising;
    when : "SE";
    timing_type : recovery_rising;
    when : "!SE";
    timing_type : removal_rising;
    when : "SE";
    timing_type : removal_rising;
    when : "!SE";
    timing_type : min_pulse_width;
    when : "CK";
    timing_type : min_pulse_width;
    when : "!CK"; }
pin (SE) {
    timing_type : hold_rising;
    when : "!R";
    timing_type : setup_rising;
    when : "!R"; }
pin (SI0,SI1,…,SIN) {
    timing_type : hold_rising;
    when : "!R&SE";
    timing_type : setup_rising;
    when : "!R&SE"; }
```

Library Power Structure

```
pin (Q0,Q1,…,QN) {
    internal_power (VDD,VNW) {
    related_pin : "CK";
    when : "!R&SE"; when : "!R&!SE";
    related_pin : "R"; when : "CK";
    when : "!CK"; }
 }
pin (CK) {
    internal_power (VDD,VNW) {
    when : "R&SE";
    when : "R&!SE";
    when : "!R&SE";
    when : "!R&!SE";
}
pin (D0,D1,…,DN) {
internal_power (VDD,VNW) {
    when : "CK&R&SE";
    when : "CK&R&!SE";
    when : "CK&!R&SE";
    when : "CK&!R&!SE";
    when : "!CK&R&SE";
    when : "!CK&R&!SE";
    when : "!CK&!R&SE";
    when : "!CK&!R&!SE";
}
```

```
pin (R) {
    internal_power (VDD,VNW) {
    when : "CK&SE";
    when : "CK&!SE";
    when : "!CK&SE";
    when : "!CK&!SE";
}
pin (SE) {
    internal_power (VDD,VNW) {
    when : "CK&R";
    when : "CK&!R";
    when : "!CK&R";
    when : "!CK&!R";
}
 pin (SI0,SI1,…,SIN) {
    internal_power (VDD,VNW) {
    when : "CK&R&SE";
    when : "CK&R&!SE";
    when : "CK&!R&SE";
    when : "CK&!R&!SE";
    when : "!CK&R&SE";
    when : "!CK&R&!SE";
    when : "!CK&!R&SE";
    when : "!CK&!R&!SE";
 }
```

Selection of mega_mode Search Settings

The Liberate `mega_mode` settings used for automatic vector search optimization have a large impact on accuracy and run time. There are separate controls for delay, constraints, and hidden power search breadth:

`set_var` mega_mode_delay `{ minimum|fanout|all }`
`set_var` mega_mode_constraint `{ minimum|fanout|all }`
`set_var` mega_mode_hidden `{ minimum|fanout|all }`

These `mega_mode_*` parameters can have the following settings:

- **`minimum`**: This setting has the smallest breadth of search, and gives the best run time by using one arbitrary combination of side pin conditions. Using this setting notifies Liberate to select a single vector for all the side pins and set the cell outputs not on the characterized path to be non-switching if possible.
- **`fanout`**: This setting has a wider breadth of search, and will cause runtime and accuracy to increase. Using this setting notifies Liberate to select all combinations of side pins that can have some effect on the measured delay or power. This includes all nodes that are switching (due to their effect on power). This mode covers all loading conditions for the delay path and provides the best accuracy. This mode might produce an exponential number of vectors if not restricted by the `-patterns` option of the define_bundle_pins command.
- **`all`**: This setting has the widest breadth of search, and will cause runtime and accuracy to be maximized. Using this setting will instruct Liberate to generate simulation vectors for all combinations of side pins. This mode is intended for maximum compatibility with non-mega (when `mega_enable` is disabled) results. For multi-bit cells, this mode produces the longest run time due to the maximum number of vectors applied. The vectors can be restricted by the `-patterns` option of the `define_bundle_pins` command.

Configuration for Automatic Optimization

In this configuration example, the search for all possible vectors is bounded by user-defined patterns that limit the combination of side input pins allowed. The pattern `{0 1 01 10}` allows only the following side pin conditions on `data` and `scan_in` pins:

- All **`0`** on side pins
- All **`1`** on side pins
- A pattern of **`0101…01`** and **`1010…10`** on side pins

```
# Optimal settings for automatic optimization of MBFF cells
set_var conditional_expression separate
set_var force_condition 5
# Allows multiple probes in one simulation, more accurate and faster
set_var constraint_probe_multiple merge
# removes bundle pins from leakage states
set_var leakage_expand_state vectors
# fastest=minimum, economical=fanout, maximum search=all
# delay default = all
set_var mega_mode_delay fanout
```

```
# constraint default = fanout
set_var mega_mode_constraint fanout
# hidden power default = fanout
set_var mega_mode_hidden fanout

# cell-specific definitions
define_cell \
    -clock { CK } \
    -input { SE SN D0 D1 D2 D3 SI0 SI1 SI2 SI3 } \
    -output { Q0 Q1 Q2 Q3 } \
    -pinlist { CK SE SN D0 D1 D2 D3 SI0 SI1 SI2 SI3 Q0 Q1 Q2 Q3 } \
    -delay delay_template_8x8  -power power_template_8x8 \
    -constraint constraint_template_4x4 \
    $cellname

# cell-specific definitions for bit-blasted bundles
define_bundle_pins -patterns {0 1 01 10} $cellname D {D0 D1 D2 D3}
define_bundle_pins -patterns {0 1 01 10} $cellname SI {SI0 SI1 SI2 SI3}
define_bundle_pins -patterns {0 1 01 10} $cellname Q {Q0 Q1 Q2 Q3}

# characterize non-state-dependent timing and power

define_bundle_pins $cellname D {D3 D2 D1 D0}
define_bundle_pins $cellname Q {Q3 Q2 Q1 Q0}
char_library …
# Generate a library with bit-blasted bundle syntax for the D and Q pins
write_library -unique_pin_data
```

Configuration for User-Defined Arcs

In this configuration example, the search for vectors is determined by user-defined arcs that specify the state of side input pins. To use this approach, it is suggested that you acquire fully expanded example templates from the Cadence Liberate support team.

```
# Optimal settings for automatic optimization of MBFF cells
set_var conditional_expression separate
set_var force_condition 5
# Allows multiple probes in one simulation, more accurate and faster
set_var constraint_probe_multiple merge
# removes bundle pins from leakage states
set_var leakage_expand_state vectors
# fastest=minimum, economical=fanout, maximum search=all
# delay default = all
set_var mega_mode_delay all
# constraint default = fanout
set_var mega_mode_constraint fanout
# hidden power default = fanout
set_var mega_mode_hidden fanout

# characterize only user-defined arcs
char_library -user_arcs_only …

# For large cells, this must be added or cell processing may not work
# But it means that the function must be defined with user_data
set_var write_logic_function 0

# Template for 2-bit DFF cell with Scan mode
define_cell \
    -clock { CP } \
    -input { D1 D2 SE SI1 SI2 } \
    -output { Q1 Q2 } \
    -pinlist { D1 D2 SE SI1 SI2 CP Q1 Q2 } \
    …
        2_BIT_SCAN_DFF

define_bundle_pins $cellname D  {D1 D2}
define_bundle_pins $cellname SI {SI1 SI2}
define_bundle_pins $cellname Q  {Q1 Q2}
```

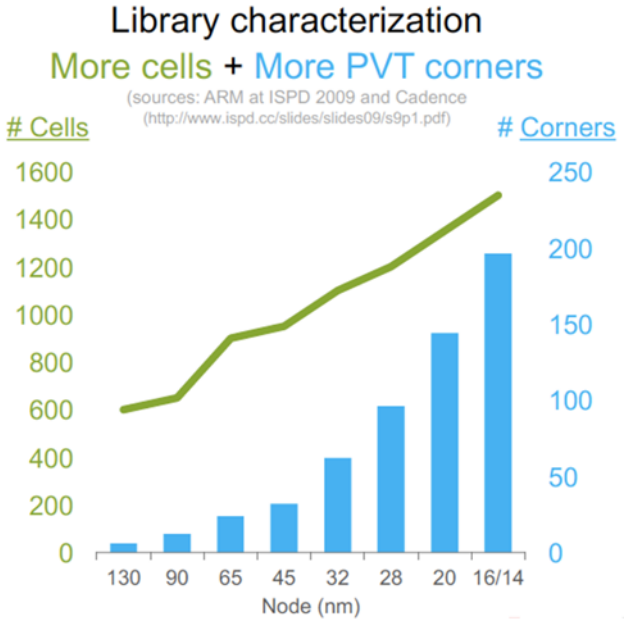
```
# Leakage states                                          DD E II C QQ
define_leakage -when " CP& D1& D2& SE& SI1& SI2" -vector {11 1 11 1 11}…
define_leakage -when " CP& D1& D2& SE& SI1&!SI2" -vector {11 1 10 1 10}…
… enumerate all legal leakage states …

# power arcs from  => D   hidden                 DD E II C QQ
define_arc -type hidden -when " CP& SE" -vector {R0 1 10 1 10} -pin D1…
define_arc -type hidden -when "!CP&!SE" -vector {0F 0 00 0 01} -pin D2…
… enumerate 16 hidden states for (rise|fall)*CP*SE*(D1|D2)
… do not enumerate Dn and Qn because extra states do not improve accuracy

# constraint arcs from CP => D              DD E II C QQ
define_arc -type setup -when "!SE" -vector {R0 0 x0 R xx} -related_pin CP -pin D1 …
define_arc -type hold  -when "!SE" -vector {0F 0 0x R xx} -related_pin CP -pin D2 …
… enumerate 8 states for (D1|D2)*(rise|fall)*(setup|hold)

# power arcs from  => SE  hidden             DD E II C QQ
define_arc -type hidden -when " CP" -vector {11 R 10 1 11} -pin SE …
define_arc -type hidden -when "!CP" -vector {11 F 10 0 10} -pin SE …
… enumerate 4 states for (D1|D2)*CP

# constraint arcs from CP => SE, choose exhaustive or worst-bit.
## Exhaustive                                           DD E II C QQ
define_arc -type setup -when "!D1& SI1&!D2&!SI2" -vector {00 R 10 R R0} -related_pin
CP -pin SE …
define_arc -type hold  -when " D2&!SI2&!D1&!SI1" -vector {01 F 00 R 0R} -related_pin
CP -pin SE …
… enumerate 16 states for D1 D2 SI1 SI2, or chose only worst case bit

# power arcs from  => SI   hidden                 DD E II C QQ
define_arc -type hidden -when " CP& SE" -vector {10 1 R0 1 00} -pin SI1…
define_arc -type hidden -when "!CP&!SE" -vector {01 0 0F 0 01} -pin SI2…
… enumerate 16 hidden states for CP*SE*(SI1|SI2)*(rise|fall)

# constraint arcs from CP => SI1           DD E II C QQ
define_arc -type setup -when "SE" -vector {x0 1 R0 R xx} -related_pin CP -pin SI1 …
define_arc -type hold  -when "SE" -vector {0x 1 0F R xx} -related_pin CP -pin SI2 …
… enumerate 8 states for (SI1|SI2)*(rise|fall)*(setup|hold)

# constraint arcs from CP => CP  min_pulse_width          DD E II C QQ
# All the same within a bit, so only need to do one per bit
```

```
define_arc -type min_pulse_width -when "D1 | SI1" -vector {10 1 00 R x0} -
related_pin CP -pin CP
… enumerate 4 states for (SI1|SI2)*(D1|D2)

# power arcs from  => CP  hidden             DD E II C QQ
# Main is SE and !SE
# Max rise_power is for 11, fall is for 00 ; 01/10 are equivalent
define_arc -type hidden -when "SE"  -vector {00 1 01 R 01} -pin CP…
define_arc -type hidden -when "!SE" -vector {01 0 00 F 01} -pin CP…
… enumerate 4 states for  SE*(rise|fall)

# delay arcs from CP => Q  non_unate rising_edge
#                              DD E II C QQ
define_arc -type edge -vector {x0 x x0 R R0} -related_pin CP -pin Q1…
define_arc -type edge -vector {0x x 0x R 0F} -related_pin CP -pin Q2…
… enumerate 4 states for  (Q1|Q2)*(rise|fall)
```

9

Using the Multi-PVT Characterization Flow of Liberate Trio

Designing at nanometer process technologies—and especially at advanced nodes (28nm and below)—poses a challenge for library development teams while characterizing a large number of Process, Voltage, and Temperature (PVT) corners.



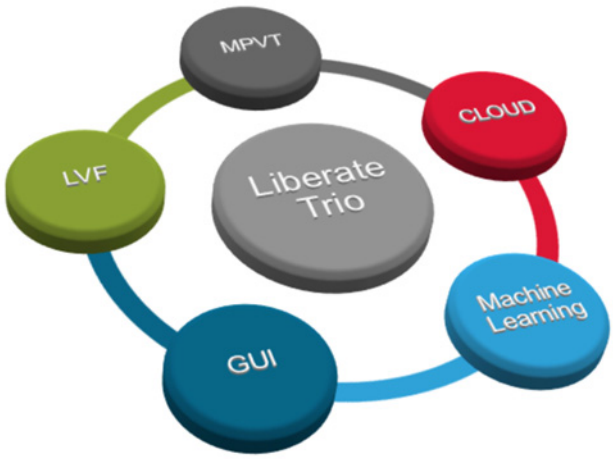
To tackle the challenges of growing library sizes in advanced nodes, users of Cadence Liberate characterization tools have developed many methods and tool configurations to build collections of Liberty and Verilog libraries.

In addition, there is now an option to use Cadence® Liberate™ Trio, a unified library characterization system that brings together features to generate and validate libraries of standard cells, custom cells, multi-bit cells, and I/Os cells.

Topics Covered

What's Cadence Liberate Trio

The cloud-ready Liberate Trio solution is scalable to thousands of cores for improved throughput and employs advanced machine-learning techniques to determine critical corners for characterization. It simplifies the challenge of dealing with an enormous collection of corners across multiple libraries.



The Liberate Trio solution brings together:

- Performance improvements with the ***multi-PVT characterization*** flow where multiple corners can be characterized in parallel.
- ***Unification of statistical and nominal modeling*** in a single characterization flow that generates Liberty Variation Format (LVF) and nominal libraries.
- ***Cloud enablement*** with massive distribution and parallelization algorithms.
- Critical corner prediction using clustering techniques from ***machine learning*** to identify and predict critical PVT corners.
- ***Real-time progress monitoring*** for each characterization job on a compute cluster through a graphical user interface (GUI) that shows status and run time for all PVT corners, cells, arcs, and data types, on every cloud machine.

Caution

Simulator-level multi-threading is not supported and recommended in Liberate Trio. The Liberate characterization tool manage CPUs better when it is controlled by clients (packet_clients) and thread (char_library -thread).

Introduction to the Multi-PVT Characterization Flow

This chapter explains the following methods of configuring the multi-PVT characterization flow to improve run time and simplify the configuration of characterization scripts used for generating large collections of libraries:

- Configuration of a single characterization script to generate many libraries.
- Restart and re-characterization of large collections of libraries done with this flow.
- Conversion of multiple existing scripts used for multi-PVT characterization flow into a single script.
- Starting and using the characterization job monitor GUI.

Important

To use the multi-PVT characterization flow, you need token licenses. To know more about the licensing requirements, refer to the Liberate Trio Licensing section in *LIBERATE Software Licensing and Configuration Guide*.

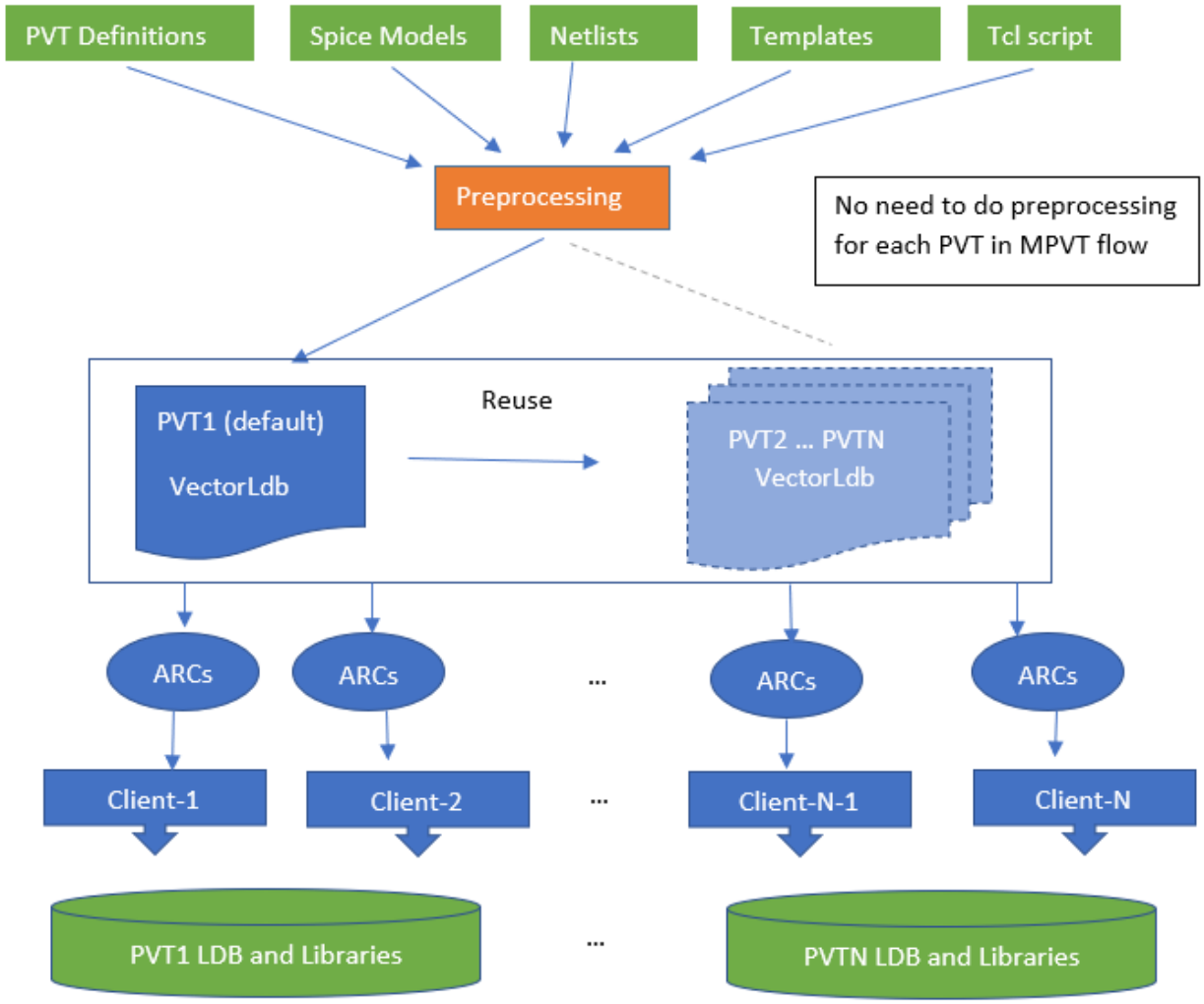
Characterization of multiple PVT corner collections of libraries could be configured with Liberate using many methods. For example:

- Construct customized scripts for each library corner and run each script in a serial flow.
- Construct a single script for all corners that uses command line parameters to pass PVT information into each script, and run a shell script to execute them in a serial flow.

Both these methods require running all functions of Liberate for each PVT corner. For example, the analysis of cell netlists to generate vectors for all arcs is done uniquely for each corner. To ensure that the structures of all libraries are identical, verbose templates are usually employed. These templates define arcs and vectors for each PVT. Then, the *Inside View* algorithm needs to be run again on each PVT corner to capture the initial conditions on all internal nodes.

The multi-PVT characterization flow eliminates redundant analysis for each PVT corner by reusing the results found on a single default corner. This ensures identical structure of libraries across all corners while saving significant run time. Further run-time savings are achieved by more efficiently running Liberate client jobs on a large number of machines. Gaps in machine uptime at the start and end of PVT corners are avoided by feeding other PVT cells

into existing Liberate clients. The library components for each cell are composed on Liberate clients to reduce total time to results.



Configuration for Multi-PVT Characterization Flow

To enable the multi-PVT characterization flow, it is necessary to run Liberate arc `packet_client` mode. For detailed information, see Arc Packet Flow.

The minimal configuration example below is taken from a working example available in the Liberate Rapid Adoption Kit (RAK) available on Cadence Online Support.

```
## Only packet_mode=arc is currently supported by MPVT flow
set_var packet_mode arc
```

```
set_var packet_clients 100
set_var rsh_cmd "local"  # or use remote job invocation such as 'bsub'

##### Define multiple PVT corners
define_pvt -default tt_1p0V_70C {
    set MODELS tt
    set VDD 1.0
    set TEMP 70
    set PROC tt_1p0V_70C
    set lib_name pvt_$PROC
}
define_pvt ff_1p1V_m25C {
    set MODELS ff
    set VDD 1.1
    set TEMP -25
    set PROC ff_1p1V_m25C
    set lib_name pvt_$PROC
}
define_pvt ss_0p9_125C {
    set MODELS ss
    set VDD 0.9
    set TEMP 125
    set PROC ss_0p9_125C
    set lib_name pvt_$PROC
}
## Define operating condition and models using active PVT
set_operating_condition -voltage ${VDD} -temp ${TEMP}
set MODEL_INCLUDE_FILE   ${SRC_DIR}/models/spectre/include_${MODELS}

## optimize Liberate clients to read netlists only of cells under test
set packet_cells [packet_slave_cells]
if {[llength $packet_cells]>0} { set cells $packet_cells }
foreach cell $cells { lappend spicefiles ${NETLIST_DIR}/${cell}.sp }
read_spice -format spectre "$MODEL_INCLUDE_FILE $spicefiles"

## Run characterization on Liberate Clients
char_library -ecsm -extsim spectre
## Save Multi-PVT database to a network location
write_ldb -overwrite ${RUN_DIR}/ldb_mpvt/${LIBNAME}.ldb

## Write libraries for all corners with unique names
```

```
## These final libraries are assembled from libraries previously generated for
## each cell and corner on the Liberate clients
set pvts [get_pvts]
foreach pvt $pvts {
    # After setting the PVT, the contents of that particular define_pvt command
    # is now available in this scope
    set_pvt $pvt
    write_library -user_data ${USERDATA} -ecsm \
        -filename ${LIB_DIR}/${lib_name}_ecsm.lib ${lib_name}_ecsm
}
```

The multi-PVT characterization flow configuration has two distinct scopes for the commands that are run on the Liberate server and those that are run on the Liberate clients. For detailed information, see Analyzing the Scope of Multi-PVT Commands on Liberate Server and Clients.

Caution

Liberate is currently limited with respect to modeling certain types of different formats within a single instance. Prime examples of these are Stage-Based CCSN and Referenced CCSN models. To avoid inconsistencies in models, it is recommended to use separate modeling Tcl files for each model to be created from an MPVT database.

Configuration for Multi-PVT Unified LVF Flow

In the releases prior to the LIBERATE 18.1 base release, nominal libraries and Liberty Variation Format (LVF) libraries were generated using separate runs with separate tools. It was necessary to merge variation data into nominal libraries to create a complete set of models in each library.

Starting from the LIBERATE 18.1 base release, in the multi-PVT characterization flow, all models can be generated in a single unified run.

```
## Sigma variation definitions
define_variation -type random { parl1 1 } parl1
define_variation -type random { parl2 1 } parl2
define_variation_group { parl1 parl2 } LOCAL_VARIATION

## Run Unified characterization on Liberate Clients
char_library -lvf -ccs -ccsn -ccsp -extsim spectre
```

```
set pvts [get_pvts]
foreach pvt $pvts {
    # generate library with LVF, and CCS timing+noise+power models
    set_pvt $pvt
    write_library -lvf -ccs -ccsn -ccsp \
        -filename ${LIB_DIR}/${lib_name}_lvf.lib ${lib_name}_lvf
}
```

Analyzing the Scope of Multi-PVT Commands on Liberate Server and Clients

In the releases previous to the LIBERATE 18.1 base release, if there was a need, commands could be constructed to be specific to the Liberate server and Liberate client. However, it was not necessary to do so, and most configuration scripts would operate regardless of attention paid to the scope of various commands.

Starting from the LIBERATE 18.1 base release, when using the multi-PVT characterization flow, it is necessary to understand and control the scope of commands. Liberate clients in this flow are now simultaneously running one specific corner each, which could be different.

Some commands are run only once on the Liberate server, while other commands are run on a constant stream of short Liberate client jobs.

Commands Run on the Liberate Server

The following commands are run on the Liberate server only one time:

- define_pvt: This command must be placed before all commands to read cell data and characterize a library.
- write_library: This command must be placed after all characterization settings and commands.
- char_library: When a library characterization job is started, the Liberate server launches all Liberate clients when the `char_library` command is first encountered.

```
# Place these commands at the top of the configuration script
define_pvt -default $pvt_name …
…
< commands and settings to load cell data and characterize >
# char_library command causes the Liberate server to begin generating vectors
# from netlists. Once this is complete, characterization jobs are
```

```
# dispatched to the Liberate clients
char_library -extsim spectre -cells $cells
write_ldb  ${LDB_NAME}.ldb
…
# The write_library command constructs the final complete libraries
write_library … -filename ${lib_name}.lib ${lib_name}
```

Note: When using multi-PVT mode, check that the arc packet mode and Bolt job distribution system are also enabled. If single PVT mode is used with the Bolt job distribution system, it is recommended to enable multi-PVT mode for a single PVT. For this, define a single PVT using the `define_pvt` command.

Commands Run on the Liberate Client

On each Liberate client, the supported commands are run during a stream of short cell characterization jobs. Each of these jobs begins for a specific PVT corner for a specific list of cells and arcs. After each short job is completed, the characterized data is saved into a temporary database network location. Library content is generated and appended to the database until all work is completed. Program control then returns to the Liberate server at the location of the char_library command.

```
# Liberate clients ignore the define_pvt commands, all PVT corners are known
…
< commands and settings to load cell data and characterize >
## Characterization for specific cells/arcs with known vectors from Liberate server
char_library -extsim spectre -cells $cells
…
# Commands that construct partial libraries for each cell/PVT corner
write_library … -filename ${lib_name}.lib ${lib_name}
```

Recharacterizing an Existing Multi-PVT Characterization Database

In the traditional single-corner Liberate flow, an existing LDB file can be read into a characterization session. A new session can then be started from the same point that the existing LDB captured.

In the multi-PVT characterization flow of Liberate Trio, the same principle applies. However, the LDB hierarchy illustrated in the Managing Multi-PVT Characterization Database Contents section can be accessed in a number of other ways. The LDB hierarchy could be read in its entirety, or portions of it may be accessed independently from the other portions.

Reading an Entire Multi-PVT Characterization Database

An entire multi-PVT characterization database can be accessed from a network location and used to write libraries or continue characterization from the point at which it was halted.

```
# Read an entire LDB with all PVT corners
read_ldb  ${LDB_NAME}.ldb

# The following command constructs complete libraries for each PVT in the database
# $lib_name is unique for each library file, using the define_pvt content
set pvts [get_pvts]
foreach pvt $pvts {
    set_pvt $pvt
    write_library -ecsm -ecsmn -user_data ${USERDATA} \
        -filename ${LIB_DIR}/${lib_name}_ecsm.lib ${lib_name}_ecsm
}
write_library … -filename ${lib_name}.lib ${lib_name}
```

Reading Only a Single Corner of a Multi-PVT Characterization Database

A single corner of a multi-PVT characterization database can be accessed from a network location and used to write libraries or continue characterization from the point at which it was halted.

```
# Read only a single corner from the LDB
read_ldb -pvt $pvt_name ${LDB_NAME}.ldb

# This command constructs a single lib for the $pvt_name corner
# $lib_name is determined from the define_pvt content for that corner
write_library -user_data ${USERDATA} \
    -filename ${LIB_DIR}/${lib_name}_ecsm.lib ${lib_name}_ecsm

# No define_pvt commands appear in this script
# Read only a single corner from the LDB
read_ldb -pvt $pvt_name ${LDB_NAME}.ldb

# This command recharacterizes a single library for the $pvt_name corner
# To make this possible, do NOT use define_pvt commands in this script
char_library -extsim spectre -cells $cells
```

Removing Cells and Recharacterizing a Specific Multi-PVT Corner

A single corner of a multi-PVT characterization database can be accessed from a network location and used to recharacterize only that specific corner. This is most useful when only a few cells need to be re-spun into the library while preserving all other cells.

```
# Read only a single PVT corner from the LDB, and remove a list of cells
# No define_pvt commands appear in this script
read_ldb -pvt $pvt_name -remove $cells ${LDB_NAME}.ldb

# This command recharacterizes a single library for the $pvt_name corner
# Only ${cells} list will be characterized, all other cells are skipped
# To make this possible, do NOT use define_pvt commands in this script
char_library -extsim spectre -cells $cells
```

Removing Cells from Multi-PVT Corners and Recharacterizing all Multi-PVT Corners

A multi-PVT characterization database can be accessed from a network location and used to recharacterize all corners for only a specific list of cells. This will recharacterize each corner for only the list of removed cells. Other existing cells will be preserved.

```
# Read the entire LDB, and remove a list of cells
# All define_pvt commands must appear in this script
read_ldb -remove $cells ${LDB_NAME}.ldb

# Only $cells list will be recharacterized for all PVT corners
char_library -extsim spectre -cells $cells
```

Enabling read_ldb-Based Recovery Characterization Flow

To run the read_ldb-based recovery characterization flow in Liberate Trio Multi-PVT (define_pvt-based) flow, set the multi_pvt_recovery_flow parameter to `1` before reading the LDB using the `read_ldb` command:

```
set_var multi_pvt_recovery_flow 1
```

Also, add the following command to point to the LDB database you want to read in the recovery session:

```
read_ldb <LDB>.ldb.gz
```

Until you run the write_ldb command, your temporary LDB database is saved to the `ldb_checkpoint_dir` directory. Therefore, you need to set the LDB database to this directory. Once the `write_ldb` command run, the LDB from the checkpoint directory moves to the `<LDB>.ldb.gz` location that can be used for recovery in the subsequent runs.

The `-overwrite` option of the `write_ldb` command is disabled for the Liberate Trio-MPVT flow to work properly. This means that every time you run the `write_ldb` command, new LDB directory (`LDB.ldb.<N>.gz`) is generated. To make it easy for the recovery flow, at the time of running the `write_ldb` command, the link to `<LDB>.ldb.gz` is always automatically updated to the latest run in the session.

The recovery flow supports the following:

- Changes in preprocessing (vector database) for any cell
- Recharacterization of specific arcs using arc IDs
- Recharacterization of specific cells and PVTs
- Changes in the characterization settings, such as parameters set using the `set_var` command, for the recovery flow

Changes in preprocessing (vector database) for any cell

In the recovery (`read_ldb`) flow, by default, the tool does not rerun preprocessing (counting) jobs. If you need to make changes in the subsequent runs that might impact the vector database, then enable rerunning of preprocessing/vector database generation using the multi_pvt_rechar_do_preprocessing parameter as shown below:

```
set_var -cell <CELL> multi_pvt_rechar_do_preprocessing 1
```

After the vector data is regenerated, the recovery flow automatically launches (forces) recharacterization of the specific cells at all PVTs. This is done to ensure that you do not use a different vector database at different PVTs.

Recharacterization of specific arcs using arc IDs

Using arc IDs, we can force recharacterization at arc level. For this, use the multi_pvt_rechar_arc_ids parameter as following:

```
set_var -cell <CELL> -pvt <PVT> multi_pvt_rechar_arc_ids [list 1 2 10]
```

The arc IDs are unique IDs assigned to each arc in the LDB. For a list, you can search the `altos_arc_index` attribute in the LDB files.

Recharacterization of specific cells and PVTs

If you need to recharacterize a specific set of cells and PVTs irrespective of whether they passed or failed in the previous session, set the multi_pvt_recovery_rechar parameter to 1. Then, you can specify the names of the required cells and PVTs using the -cell and -pvt options with the set_var command to enable:

- Recharacterization of specific cells at few or all PVTs
- Recharacterization of a specific few or all cells for a specific PVT

For example:

```
set_var multi_pvt_recovery_flow 1
set_var –cell * -pvt *125c multi_pvt_recovery_rechar 1 ; # all cells at PVTs having 125c in the PVT name set using the define_pvt command
set_var –cell DFF* multi_pvt_recovery_rechar 1 ; # cell names starting with DFF at all PVTs
read_ldb <LDB>.ldb.gz
```

Changes in the characterization settings

In the recovery flow, you have the choice to change any parameter or command settings that impact the characterization run. Any changes in the characterization settings have an impact on various arcs, cells, and PVTs depending on whether the changes you made are global or specific to certain arcs, cells, and PVTs.

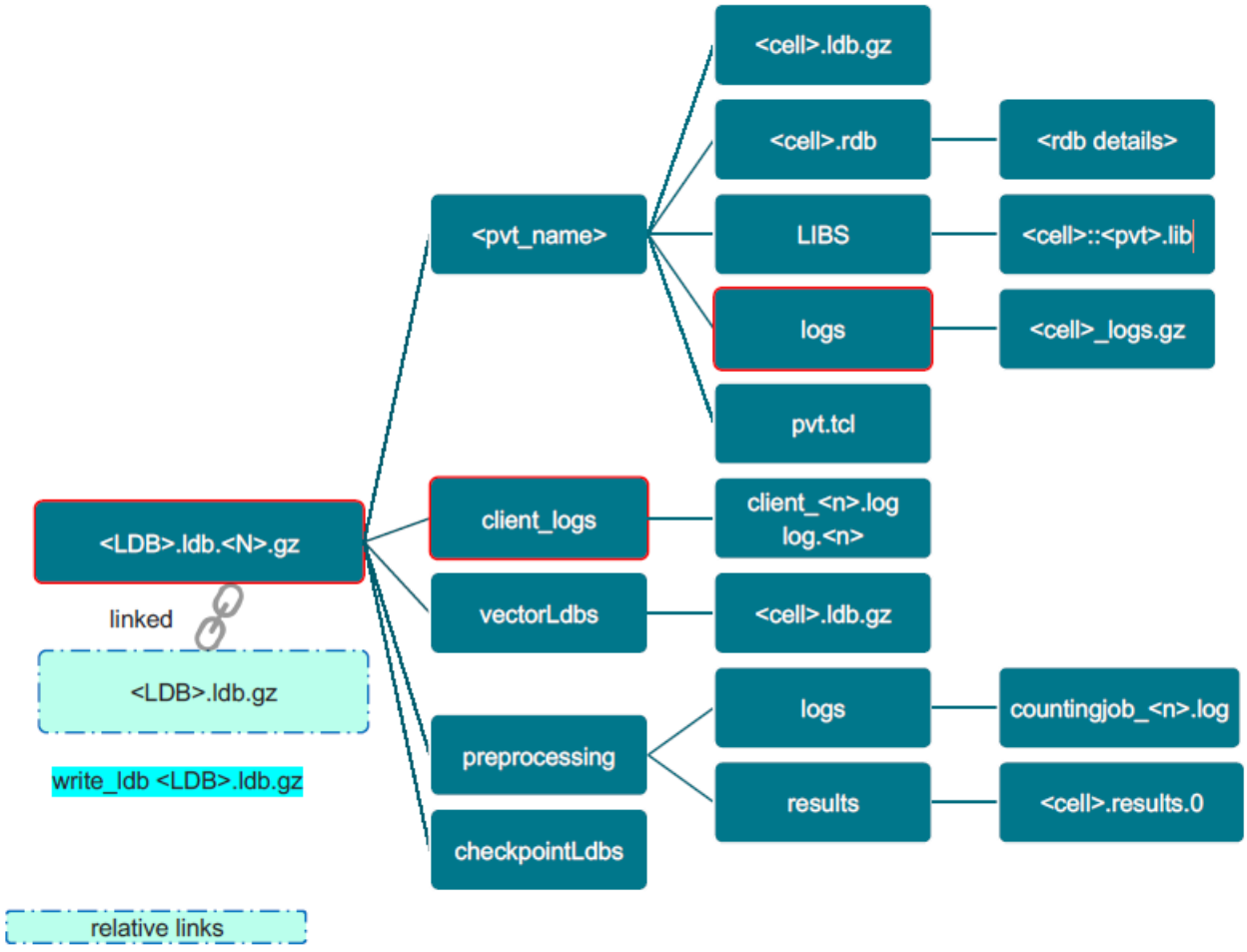
To ensure that your changes are applied to the arcs, cells, and PVTs for which you want to force recharacterization, set the multi_pvt_recovery_rechar parameter to 1. These modified settings also apply to the arcs, cells, and PVTs that were not successful in the previous runs and are expected to be rerun in the recovery flow.

For example:

```
set_var multi_pvt_recovery_flow 1
set_var –cell DFFQRS* -pvt *125c multi_pvt_recovery_rechar 1
read_ldb <LDB>.ldb.gz
set_var leakage_merge_state 2 ; # applicable to all cells at all PVTs matching *125c and failed leakage arcs for
; # any cells at other PVTs in previous run
set_var -cell DFFQRSX1 constraint_delay_degrade 0.20 ; # applicable to this specific cell at all PVTs being matching *125c and failed
; # constraint arc for this cell in previous run at other PVTs
```

Managing Multi-PVT Characterization Database Contents

As the multi-PVT characterization database is generated, the data begins to inflate with completed timing, power, noise, and sensitivity data. This data includes log files from each Liberate client, single-cell libraries for each completed cell within each corner, and partially completed data from all cells and arcs. At the end of a fresh characterization run, a hierarchy similar to the one illustrated below is created:



In the multi-PVT characterization flow, if the `-overwrite` option is specified with the `write_ldb` command, then it is ignored.

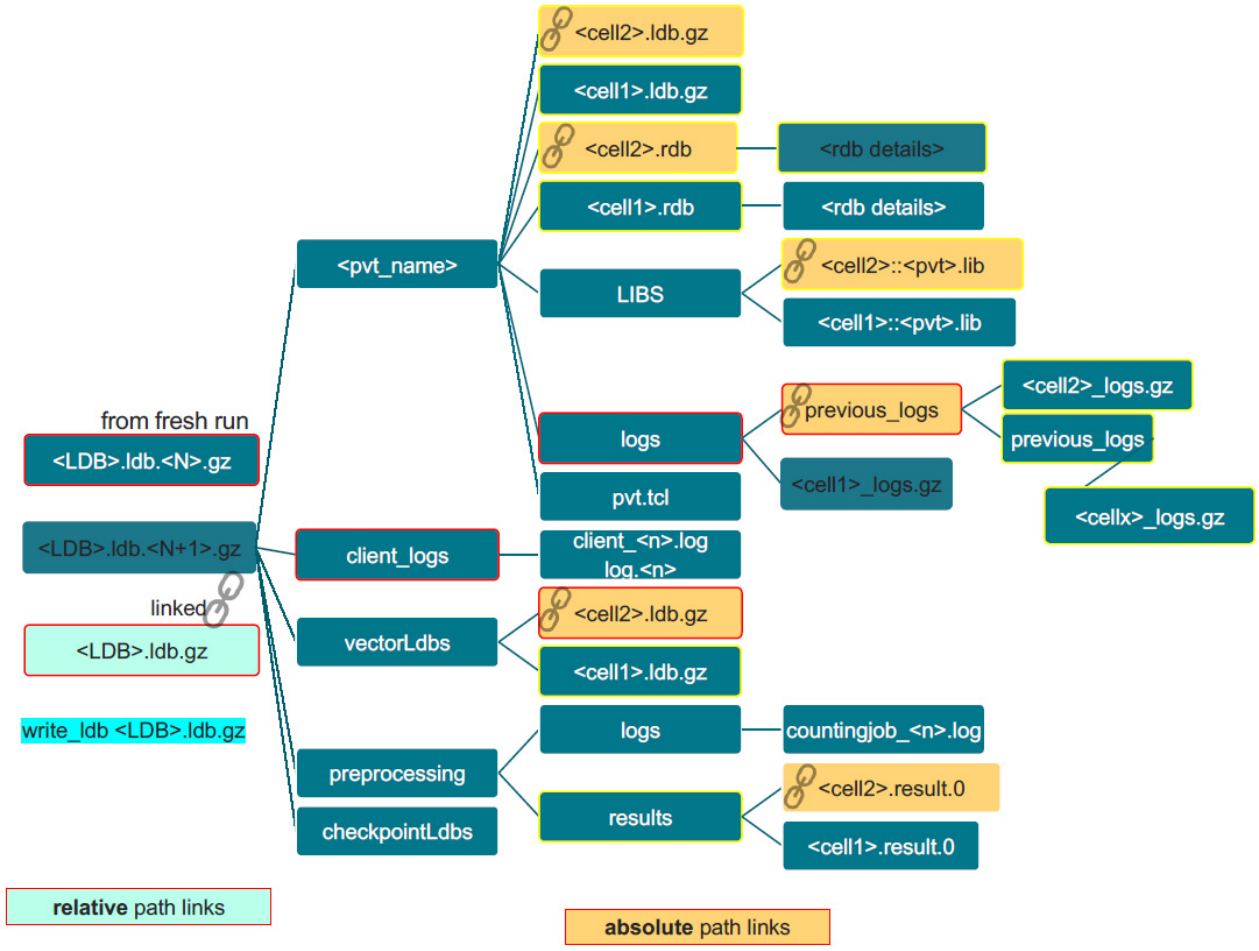
Unique LDB directories are generated with names of the format `<LDB>.ldb.<N>.gz`. The latest LDB directory is always linked to `<LDB>.ldb.gz`. This linking ensures that the `read_ldb <LDB>.ldb.gz` recovery flow always uses data from the latest LDB directory.

In addition, `<pvt_name>` is automatically added to `extsim_deck_dir` to avoid it getting overridden.

Caution

The entire data in the LDB is in human readable text format. Altering this data in any way could cause the database to become unusable by Liberate tools. Any editing of this data should be done using only the Liberate API and Liberate DataBase eXplorer tools.

If the read_ldb-based recovery flow is enabled using the multi_pvt_recovery_flow parameter, a hierarchy similar to the one illustrated below is created:

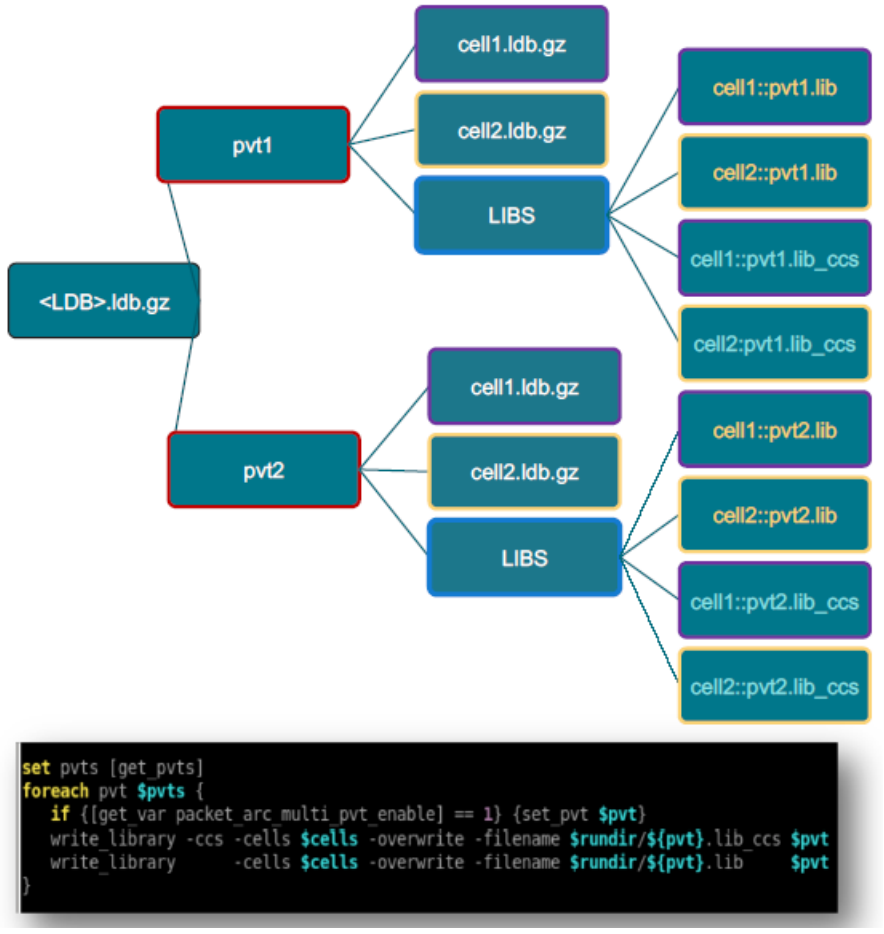


In this recovery flow, data is read from the linked LDB directory (`LDB.ldb.<N>.gz`), where N is the number for the previous run. After the `read_ldb` session is finished, a new LDB directory `LDB.ldb.<N+1>.gz` is created when the write_ldb command is run and now this becomes latest LDB directory that is linked to `LDB.ldb.gz`.

Vector database, cell-level LDBs and log files for each PVT from the previous sessions are linked to the current session. This is done to ensure that all relevant and important details are available in the current session. In addition, cell-level logs from the previous session are linked in the `previous_logs` directory under the `<pvt>/logs` directory of the current session.

Library Modeling in Multi-PVT Characterization Flow

The following figure illustrates how a library is modeled in the Multi-PVT characterization flow:



Library Modeling on the Client Side

A cell-level library (`<pvt>/LIBS/<cell>::<pvt>.lib`) is modeled on the client during the same assembly job after the RDB to LDB assembly has been done. The cell-level modeling happens during the char_library phase on the client. The write_library command in the

characterization script is run on the client after the `char_library` and write_ldb commands. By the end of the `char_library` run, good LDBs and LIBs are generated for all cells and PVTs.

When the `write_library` command is run, a summary of the modeled library is printed in the following format:

```
Modeling statistics:
Number of passing cells {<pvt_name>} = 1
Number of failing cells {<pvt_name>} = 0
List of failing cells {<pvt_name>} = {}
```

Library Modeling on the Server Side

If the characterization run has completed in a session and the cell-level libraries were generated on the client, then on the server side, the `write_library` command internally runs the append_library command. Therefore, in the `read_ldb`-based recovery characterization flow, changing the value of `-filename` option in the `write_library` command is not recommended. If the `-filename` option in this recovery flow is different from the fresh characterization flow, then the `write_library` command is run based on the read LDB and a summary is printed in the following format:

```
Modeling statistics:
Number of passing cells {<pvt_name>} = 1
Number of failing cells {<pvt_name>} = 0
List of failing cells {<pvt_name>} = {}
```

Distributed Modeling Flow

Consider that the characterized LDBs you generated are good, but you now want to generate another one with a different combination for library files. So, you want to change some modeling-related parameters or commands and want that the new model reflects the changes. For example, you generated the *Reference CCSN + C1CN* model in the fresh characterization session and want to generate the *Stage CCSN + C1N2* model in the subsequent session. Then, change the filename specified with the `write_library -filename` option to avoid overriding original the library files. However, this approach is not recommended in the read_ldb-based recovery flow. This is where you can use the distributed modeling flow that is based on the read_ldb flow to model the new flavor of libraries or to make changes in the setup that impacts only modeling, such as `set_attribute`, `user_data` and so on, in the distribution mode.

In this flow, the cell-level modeling is distributed on the clients as assembly jobs where the LDB will be read and cell-level libraries will be generated. Once that is done, then on the server side, the two set of libraries will be appended together automatically to get a library-

level `.lib` file. To enable this, set the bolt_post_char_command_distribution parameter to `1` before running the `read_ldb` command.

Important

The distributed modeling flow requires full and valid characterization script including `char_library` for the Liberate Trio Multi-PVT flow.

Converting Existing Liberate Scripts for Multi-PVT Characterization Flow

Liberate users use many configuration styles. Some are simple to convert to multi-PVT characterization flow, especially those which use a common top-level script for all corner characterization. Configuration styles that use different script files for each corner require more effort, but it is still possible to construct a single top-level configuration script to control them all.

Converting a Flow with Single Characterization Configuration File to Multi-PVT Characterization Database

One method to generate a large number of corners is to use a shell script to run a series of Liberate jobs. By passing command line parameters into the Liberate job, each invocation can use a common, `char_single.tcl`, configuration script to set the PVT corners for each characterization job. The characterization job then creates a separate LDB for each corner.

Here is a simplified example of such a flow:

```
<< Content of run_char.sh shell script >>


#!/bin/bash
# Define all PVT corners to be characterized by running
# Liberate multiple times with PVTs defined as command line parameters
liberate char_single.tcl process=tt vdd=1.0 temp=70 lib_name=tt_1p0V_70C 2>&1|tee
char_tt.log
liberate char_single.tcl process=ff vdd=1.1 temp=-25 lib_name=ff_1p1V_m25C
2>&1|tee char_ff.log
liberate char_single.tcl process=ss vdd=0.9 temp=125 lib_name=ss_0p9_125C 2>&1|tee
char_ss.log


<< Partial content of Liberate char_single.tcl script >>
# Use command line parameters to define a unique PVT corner
source ${SRC_DIR}/tcl/cmdline_params.tcl
```

```
set PROCESS              $process
set VDD                  $vdd
set TEMP                 $temp
set LIBNAME              pvt_$lib_name
set MODEL_INCLUDE_FILE   ${SRC_DIR}/models/spectre/include_${PROCESS}
read_spice -format spectre "$MODEL_INCLUDE_FILE $spicefiles"
set_operating_condition -voltage ${VDD} -temp ${TEMP}
char_library -extsim spectre -cell $cells
write_library -ecsm -ecsmn -filename ${LIBDIR}/${LIBNAME}_ecsm.lib ecsm
```

This single configuration script flow can be converted to a multi-PVT characterization flow by eliminating the `run_char.sh` shell script entirely, and defining all PVT conditions directly in a new `char_mpvt.tcl` Liberate script. Here is a simplified example of that converted script:

```
<< Partial content of Liberate char_mpvt.tcl script >>
# Define all PVT corners to be characterized in a single Liberate run
define_pvt -default tt_1p0V_70C {
    set MODELS tt
    set VDD 1.0
    set TEMP 70
    set PROC tt_1p0V_70C
    set lib_name pvt_$PROC
}
define_pvt ff_1p1V_m25C {
    set MODELS ff
    set VDD 1.1
    set TEMP -25
    set PROC ff_1p1V_m25C
    set lib_name pvt_$PROC
}
define_pvt ss_0p9_125C {
     set MODELS ss
     set VDD 0.9
     set TEMP 125
     set PROC ss_0p9_125C
     set lib_name pvt_$PROC
}
set MODEL_INCLUDE_FILE   ${SRC_DIR}/models/spectre/include_${MODELS} read_spice -
format spectre "$MODEL_INCLUDE_FILE $spicefiles"
set_operating_condition -voltage ${VDD} -temp ${TEMP}
char_library -extsim spectre -cell $cells
# Write unique libs for all PVT corners.  The set_pvt command puts the
```

```
# $lib_name into the scope of the write_library command
foreach pvt $pvts {
    set_pvt $pvt
    write_library -filename ${LIBDIR}/${lib_name}_ecsm.lib ecsm
}
```

Converting a Flow with Multiple Configuration Files to Multi-PVT Characterization Database

Another method to generate a large number of corners is to generate unique characterization run scripts for each PVT. Each script can contain settings and commands specific to that PVT corner. Each characterization job then creates a separate LDB database for each corner. Here is a simplified example of such a flow:

```
<< Content of run_char.sh shell script >>

#!/bin/bash
# Run Liberate multiple times with PVTs defined within unique scripts
liberate char_tt_70c_1p0.tcl 2>&1|tee char_tt.log
liberate char_ff_m70c_1p1.tcl 2>&1|tee char_ff.log
liberate char_ss_125c_0p9.tcl 2>&1|tee char_ss.log

<< Partial content of Content of Liberate char_mpvt.tcl script >>
# Define all PVT corners to be characterized in a single Liberate run
define_pvt …

# Use existing scripts to characterize this PVT on the Liberate Client
set this_pvt [get_var active_operating_pvt]
source $script_dir/char_${this_pvt}.tcl
char_library -extsim spectre -cell $cells

# Use existing scripts to write PVT-specific libraries
set pvts [get_pvts]
foreach pvt $pvts {
    set_pvt $pvt
    source $rundir/write_ecsm_$pvt.tcl
}
```

Note: In this converted flow, all hierarchical Tcl commands need to be modified such that the current PVT content is used to select the correct files and settings to include from the existing collection of scripts.

Using Custom Definitions in Multi-PVT Characterization Scripts

Requirements often arise to customize some settings and commands for each PVT corner. This can be accomplished in the multi-PVT characterization flow by writing a code that is conditional on the current PVT corners on the Liberate client. For example, this is a way to control the way in which setup constraints are captured on different corners:

```
# Get the PVT name string from the define_pvt present in the current scope
set this_pvt [get_var active_operating_pvt]

# Change the constraint_criteria for each PVT corner
if {[string match $this_pvt "ss_0p9_125C"]} {
    set_var constraint_delay_degrade 0.2
    # also test for slew degradation on setup constraints for this corner
    set_constraint_criteria -type setup -slew_degrade 0.2
elseif {[string match $this_pvt "ff_1p1V_m25C"]} {
    set_var constraint_delay_degrade 0.15
else {
# Use default for all other PVTs
    set_var constraint_delay_degrade 0.10
}
```

Using User-Defined Overrides to Control Characterization

Liberate Trio supports the following two user-defined overrides while the characterization run is in progress:

- `EXIT_CHARACTERIZATION`

  In the `altos*ldb.gz` directory, if you create an `EXIT_CHARACTERIZATION` file, which can also be an empty file, Liberate exits the characterization and prints a summary of passed and failed cells. Thereafter, Liberate proceeds to the next set of commands such as `write_library` and so on.

  An information message such as following is generated:

  ```
  INFO (LIB-481): (user override): Detected an 'EXIT_CHARACTERIZATION' file in
  the altos.*.ldb.gz directory. Stopping characterization and proceeding to the
  next command, if any is found in the Tcl script.
  ```

- `EXIT_TOOL`

In the `altos*ldb.gz` directory, if you create an `EXIT_TOOL` file, which can also be an empty file, Liberate stops all executable that are currently running and exits immediately.

An information message such as following is generated:

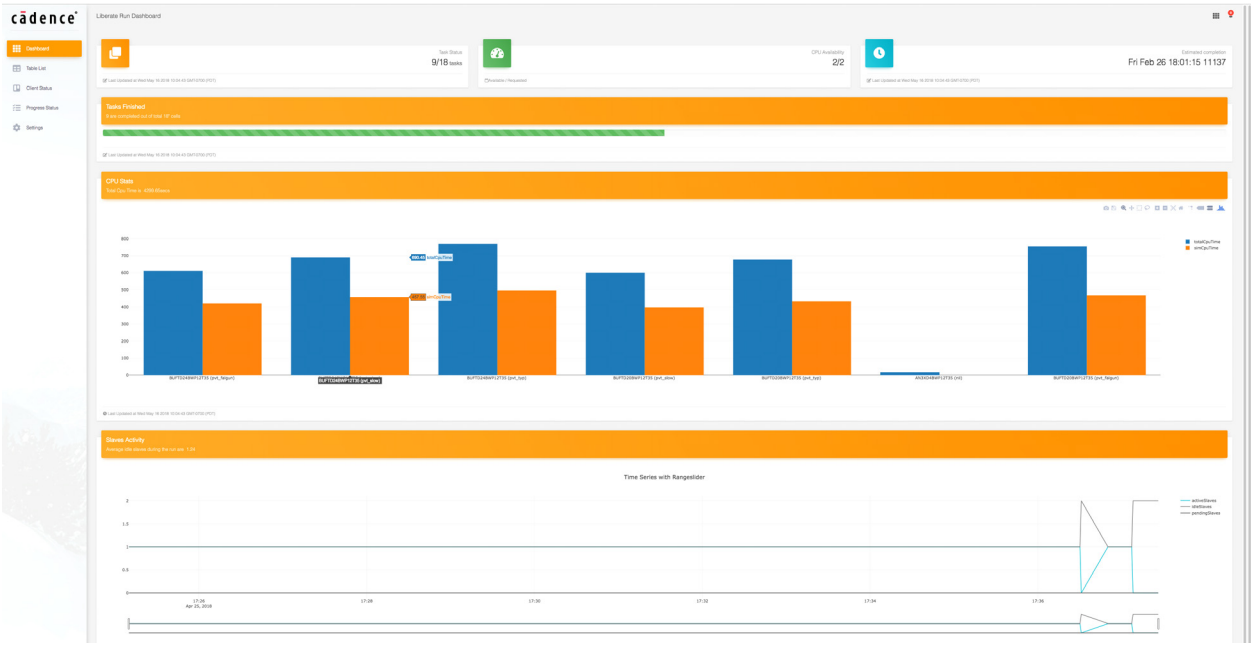
```
INFO (LIB-481): (user override): Detected an 'EXIT_TOOL' file in the
altos.*.ldb.gz directory. Stopping characterization and exiting the tool.
```

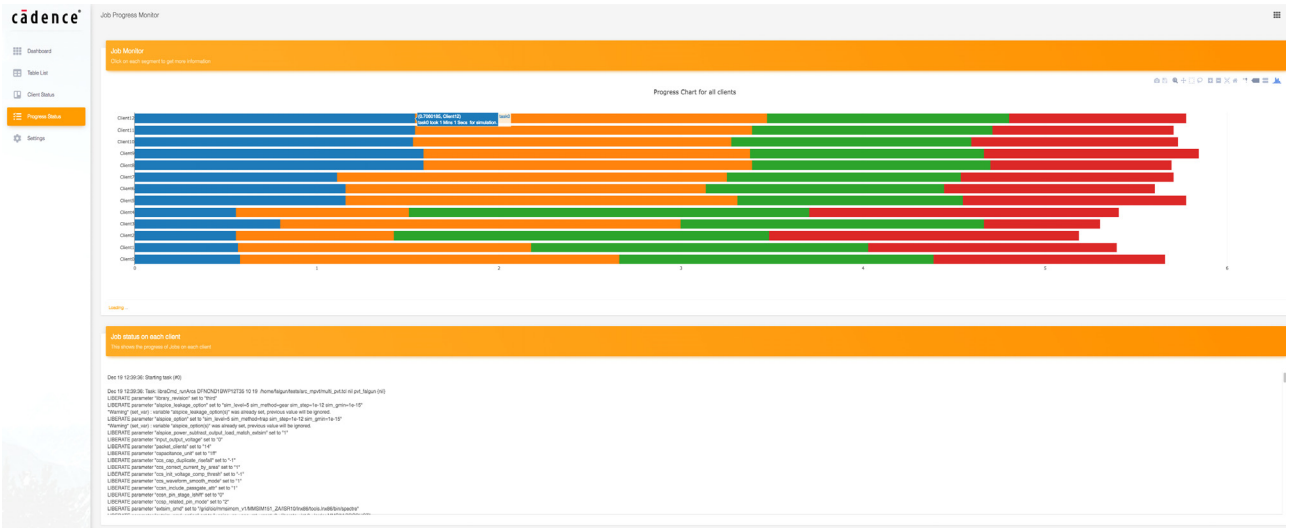
Using the Liberate Characterization Monitor

The multi-PVT characterization flow supports the use of Liberate run-time performance and results monitoring in the Liberate Characterization Monitor. This GUI interface monitors all PVT corners and cells running on a distributed CPU network. To enable this GUI interface, add the following commands into the characterization script:

```
# Launch the Liberate GUI job monitoring cockpit
set_var web_server_enable 1

# Send job notifications to this email address
set_var packet_arc_notification_list user@company.com

# Set maximum number of email notifications
set_var packet_arc_notification_limit 100

# Set interval between notification emails
set_var packet_arc_notification_interval 900
```

The following images illustrate the Liberate Characterization Monitor as it reports run-time performance, status, and results.





To track the progress and overall health of jobs running on Liberate clients, you can run the Liberate Characterization Monitor in any web browser or in a desktop application such as Safari.

Clicking the *Table List* menu in the left pane of the Liberate Characterization Monitor displays the details of different cells and how long they took to run. The tabular structure of the displayed report lets you sort the contents based on the column header you click.

Clicking the *Client Status* menu displays the existing jobs that are currently running to monitor their progress at any given moment the Client Log section of the GUI interface.

In addition, clicking the *Progress Status* menu lets you to view the overall health of the characterization run. This color-coded view shows hundreds of tasks that are being run on various Liberate client showing the relative run time of each task. There are hundreds of tasks in this view. You can probe further by clicking a specific task that displays detailed information related to it.

The *Client Activity* chart of the Liberate Characterization Monitor shows live information about how Liberate is scheduling the jobs and using the system resources. This chart information such as following: *clientsTotal*, *pendingClients*, *aliveClients*, *closedClients*, *exitedClients*, *jobsSubmitted*, *jobsRemaining*, *countingJobs*, *oneshotjobs*, *simulationJobs*, *assemblyJobs*, and *cellsRunning*.

A

Truth Table Format

Notation recognized in Liberate Truth Table format:

```
#   -- comment line, when # appears at the beginning of line
*   -- command line, when * appears at the beginning of line
```

If '#' appears as the first non-white character of a line, then this line is a comment and all text in this line is ignored.

Truth Table Format – Basic

The following are basic truth table commands:

ASYNC_PINS={Pin1 Pin2...}

Specifies the **define_cell async** pins that are included in the **INTERNAL_PINS**. If there is a conflict between the **ASYNC_PINS** and the truth table specified pins, Liberate will honor the information from the truth table.

For example:

```
* ASYNC_PINS={A B C}
```

BIDI_PINS={Pin1 Pin2...}

Specifies the **define_cell bidi** pins that are included in the **INTERNAL_PINS**. If there is a conflict between the **BIDI_PINS** and the truth table specified pins, Liberate will honor the information from the truth table.

For example, if pin PAD is specified as:

```
* BIDI_PINS={PAD}
```

but in the truth table, we have pin PAD defined as an output:

```
* TABLE = A  GZ  T0 SR0 @ PAD
```

```
0   0   0  1  @  0
1   0   0  1  @  1
-   1   0  1  @  Z
```

the pin PAD will be added to the output option of the **define_cell** command:

```
define_cell -output {PAD} ...
```

CELL=<cell_name>

Specifies the cell name. A cell table starts with this command and ends with the **TABLE_END**. All commands in this range will apply to this cell only.

CELL_END

This is the same as **TABLE_END**. If it is used at the end of the last table definition, it can provide readability by indicating the end of the cell definition.

CLOCK_PINS={Pin1 Pin2...}

Specifies the **define_cell clock** pins that are included in the **INTERNAL_PINS**. If there is a conflict between the **CLOCK_PINS** and the truth table specified pins, Liberate will honor the information from the truth table.

CONSTRAINT_TEM=<name>

Specifies the constraint template name to be inserted to define_cell.

DELAY_TEM=<name>

Specifies the delay template name to be inserted to define_cell

DUAL_IN= {<pin>:<dual_pin>}
DUAL_IO= {<pin>:<dual_pin>}
DUAL_OUT={<pin>:<dual_pin>}

Specifies a pair of pins that are dual-related pins. They must be of the same type (both are output or both are input) and have different (non-unate) toggle direction. (For example, if one pin transitions R -> F, then the dual pin must transition F -> R.)

The two dual pins must switch at the same time.

```
# Example
* DUAL_OUT={PADP:PADN}
* DUAL_IN={IPADP:IPADN}
* DUAL_IO={BPADP:BPADN}
```

ENABLE_PINS={out_pin:enablePin[,sideInputPin ,..] out_pin:enablePin[,..]...

Specifies the enable pins of a tri-state output pin. Currently, Liberate only recognizes one enable pin per tri-state in/output pin (A tri-state bidi pin has to have a Z state, and it appears in both the input and output columns of the truth table). This command is used to help specify two or more enable pins. Multiple commands can be used, or a single command can have one or more lines. (In that case, use curly braces to group pin the list.) Example:

```
# if A1 -> PAD1 has enable pin HHV and EN1, and A2->PAD2  has
# enable pins HHV and EN2, then the following can be used:
* ENABLE_PINS=PAD1:EN1,HHV
* ENABLE_PINS=PAD2:EN2,HHV
    or
* ENABLE_PINS={PAD1:EN1,HHV  PAD2:EN2,HHV}
```

IGNORE_ARC=< pin:related_pin [, Related_Pin ...] >

Specifies to Liberate which arc(s) (Related_Pin to Pin) should be ignored. Multiple related pins can be grouped together with ',' for the same output pin. This command is useful if there are ambiguous arcs in the truth table that are not required to be characterized with a define_arc command. Example :

```
* IGNORE_ARC={PAD:PE,IE AI:IE DI:IE}
```

INPUT_PINS={Pin1 Pin2...}

Specifies the **define_cell input** pins that are included in the **INTERNAL_PINS**. If there is a conflict between the **INPUT_PINS** and the truth table specified pins, Liberate will honor the information from the truth table.

INTERNAL_PINS={Pin1 Pin2...}
INTERNAL_PINS={Pin1:value Pin2:value...}
INTERNAL_PINS={Pin1:value Pin2:value Pin3 Pin4 Pin5:value...}

The **INTERNAL_PINS** command has been modified to set pins which are not listed in the truth table but need to be added to the **define_cell pin_list**. The user can set values for these pins. The values will be added to the **define_arc vector** option. Only one value per pin is

supported at this time. Supported values are: **0**, **1**, and **X**. Any pin in the list without a value will be set to **X**. Example:

```
* INTERNAL_PINS={UP_N:0 UP_P:0 SCANOUT:X Y1 Y2 PI:1 PO:1}
* TABLE=  A  GZ  T0 SR0 @ PAD
          0   0  0   1  @ 0
          1   0  0   1  @ 1
          -   1  0   1  @ Z
```

This would be the resulting define_arc command:

```
define_arc \
    -vector "F001F00XXX11" \
    -when "!GZ&!T0&SR0" \
    -related_pin A \
    -pin PAD \
    Mycell
```

Note: Any pin defined as an INTERNAL_PINS in the truth table command must also be specified as either an INPUT_PINS, OUTPUT_PINS, BIDI_PINS, CLOCK_PINS, ASYNC_PINS or SLEEP_PINS. Otherwise the internal pin will be ignored and the simulation run might fail.

LEAKAGE_ENUM_PINS= {Pin1, Pin2, ...}

Defines the pins to be enumerated for generating define_leakage 'when' states. If not specified, then all input pins and tri-state pins are utilized. By default, up to 8 pins are accepted for enumeration. This is controlled by the parameter **max_leakage_vector**, which defaults to 256 leakage states.

For example:

```
* LEAKAGE_ENUM_PINS={PAD data_out data_out_en st_en}
```

MAP_IGNORED_BIT=<character>

Maps a "-" to another supported character in the **define_arc** vector. The valid data is a single character of **0**, **1**, **x**, or **X** . For example, a vector of "0-0-RR" becomes "0X0XRR" in the **define_arc** command when the following is used:

```
* MAP_IGNORED_BIT=X
```

The default is "X" (map to an "X").

Note: The above mapping is applied only to input fields.

MODE = < ENUM_ALL | ENUM_ARC | ENUM_LEAKAGE | NO_ENUM | STD >

ENUM_ALL	Liberate will enumerate all pins or only those pins specified by the **LEAKAGE_ENUM_PINS** command to generate define_leakage commands in the template. It will also enumerate all pins to generate define_arc commands in the output template.
ENUM_ARC	Liberate will follow the truth table entries to generate define_leakage commands in the template. All pins will be enumerated to generate define_arc commands.
ENUM_LEAKAGE	Liberate will enumerate all pins or only those pins specified by the **LEAKAGE_ENUM_PINS** command to generate define_leakage commands in the template. The define_arc commands will follow the truth table entries.
NO_ENUM | STD	Liberate will follow the truth table entries to generate define_leakage and define_arc commands in the output template file. (Default.) Note: **STD** and **NO_ENUM** are equivalent.

OUTPUT_PINS={Pin1 Pin2...}

Specifies the **define_cell output** pins that are included in the **INTERNAL_PINS**. If there is a conflict between the **OUTPUT_PINS** and the truth table specified pins, Liberate will honor the information from the truth table.

PINLOAD_DIR=dir

Specifies the pin load direction in pin load template, where dir is "U", "D" and "B" (up, down, both). If a direction is not specified, no load will be added. (Default)

PINPAIR=<pin1:pin2,...>

Identifies the differential pin pairs. Multiple pin pairs can be specified when separated by a comma (",").

POWER_TEM=name

Specifies the power template name to be inserted to **define_cell**.

PULLUP=<pins>
PULLDOWN=<pins>

Specifies the pins that can only pull-up or pull-down. For the pull-up/down pins, only the pin-caps are characterized. Note: These commands are not supported at this time and will be supported in a future release.

PULLUP_RES=<resistance>
PULLDOWN_RES=<resistance>

Specifies the pull-up/down resistance (float, in Ohms) for the pull-up/down pins. If this is not specified, the resistance will not be output.

PULLUP_VOLT=<voltage>

Specifies the pull-up voltage (float, in Volts)

RELATED_PINS={Pin:Related_Pin [,Related_Pin ...] }

Specifies to Liberate which related pin(s) are associated with a given output pin. Multiple related pins can be grouped together with ',' for the same output pin. This command is useful if there are ambiguous arc's in the truth table for which Liberate might fail to find a related pin for an output pin.

Example:

```
* RELATED_PINS={PAD:DO AI:DO DI:DO}
```

SERIES_RES=<resistance>

Specifies the series connected resistance (float, in Ohms) to add to the bi-directional pin.

SI_IM_TEM=name

Specifies the SI Immunity template name to be inserted to define_cell

TABLE=Inputs@Outputs
<state_values>
TABLE_END

Specifies the truth table. The Input and Output pin arcs are specified as input pins, a separator, then the output pins. The pin names and the values must be separated by whitespace. The values are specified one vector at a time. The table continues until the command **TABLE_END** is encountered. Bi-directional pins are specified as both inputs and outputs.

Example:

```
* TABLE OEN I PAD @ PAD C
            1  -  @  1  1
* TABLE_END
```

TABLE_SEP=<value>

Specifies the default table separator value.

(Default: '@'). Specify this command at the beginning of the input file (not inside any *CELL) to specify separator globally.

WRITE_HIDDEN_POWER = < 0 | 1 >

Controls the output of hidden power arcs into the template. Hidden arcs are arcs where an input toggles, but no output toggles. Default is 0 (do not write hidden power.)

WRITE_WHEN = < 0 | 1 >

Adds the `-when` option to the `define_arc` commands. This option can be applied globally when used before the CELL command, or locally for individual cells when used after the CELL command. Default = 0.

Note: If the '-when' is present in a `define_arc` command, it will be carried into the lib.

Example 1:

```
* CELL= MUXI31
* MODE=ENUM
* WRITE_WHEN=1
* LEAKAGE_ENUM_PINS={S0 S1 S2}
* TABLE= I0 I1 I2 S0 S1 S2 @ X
         1  x  x  1  0  0  @ 0
```

```
          0  x  x  1  0  0  @ 1
* TABLE_END
```

Example 2:

In file `gen.tcl`:

```
read_truth_table mux.tt
write_template -truth_table template_mux
```

In file `mux.tt`:

```
* CELL= MUXI31
* MODE= ENUM
* WRITE_WHEN=1
* LEAKAGE_ENUM_PINS={S0 S1 S2}
* TABLE= I0 I1 I2 S0 S1 S2 @ X
          1  x  x  1  0  0  @ 0
          0  x  x  1  0  0  @ 1
          x  1  x  0  1  0  @ 0
          x  0  x  0  1  0  @ 1
          x  x  1  0  0  1  @ 0
          x  x  0  0  0  1  @ 1
* TABLE_END
* CELL_END
```

The generated `define_leakage` and `define_arc` commands in `template_mux.tcl` file:

```
# define_leakage for 3 pins :
# S0 S1 S2
define_leakage -when "S0 S1 S2" {MUXI31}
define_leakage -when "!S0 S1 S2" {MUXI31}
define_leakage -when "S0 !S1 S2" {MUXI31}
define_leakage -when "!S0 !S1 S2" {MUXI31}
define_leakage -when "S0 S1 !S2" {MUXI31}
define_leakage -when "!S0 S1 !S2" {MUXI31}
define_leakage -when "S0 !S1 !S2" {MUXI31}
define_leakage -when "!S0 !S1 !S2" {MUXI31}

# Generate define_arc by enumerating all possible vector combination.
# Pins : I0 I1 I2 S0 S1 S2 X
define_arc -vector "FXX100R" -when "S0&!S1&!S2" -related_pin {I0}    -pin X
MUXI31
```

```
define_arc -vector "10XFR0R" -when "I0&!I1&!S2" -related_pin {S0 S1} -pin X
MUXI31
define_arc -vector "1X0F0RR" -when "I0&!I2&!S1" -related_pin {S0 S2} -pin X
MUXI31
define_arc -vector "RXX100F" -when "S0&!S1&!S2" -related_pin {I0}    -pin X
MUXI31
define_arc -vector "01XFR0F" -when "!I0&I1&!S2" -related_pin {S0 S1} -pin X
MUXI31
define_arc -vector "0X1F0RF" -when "!I0&I2&!S1" -related_pin {S0 S2} -pin X
MUXI31
define_arc -vector "01XRF0R" -when "!I0&I1&!S2" -related_pin {S0 S1} -pin X
MUXI31
define_arc -vector "XFX010R" -when "!S0&S1&!S2" -related_pin {I1}    -pin X
MUXI31
define_arc -vector "X100FRR" -when "I1&!I2&!S0" -related_pin {S1 S2} -pin X
MUXI31
define_arc -vector "10XRF0F" -when "I0&!I1&!S2" -related_pin {S0 S1} -pin X
MUXI31
define_arc -vector "XRX010F" -when "!S0&S1&!S2" -related_pin {I1}    -pin X
MUXI31
define_arc -vector "X010FRF" -when "!I1&I2&!S0" -related_pin {S1 S2} -pin X
MUXI31
define_arc -vector "0X1R0FR" -when "!I0&I2&!S1" -related_pin {S0 S2} -pin X
MUXI31
define_arc -vector "X010RFR" -when "!I1&I2&!S0" -related_pin {S1 S2} -pin X
MUXI31
define_arc -vector "XXF001R" -when "!S0&!S1&S2" -related_pin {I2}    -pin X
MUXI31
define_arc -vector "1X0R0FF" -when "I0&!I2&!S1" -related_pin {S0 S2} -pin X
MUXI31
define_arc -vector "X100RFF" -when "I1&!I2&!S0" -related_pin {S1 S2} -pin X
MUXI31
define_arc -vector "XXR001F" -when "!S0&!S1&S2" -related_pin {I2}    -pin X
MUXI31
```

Liberate Truth Table Format

Input Field

```
'0' = Low
'L' = low (higher or equal to 0. Currently is treated as '0')
'1' = High
'H' = high (lower or equal to 1. Currently is treated as '1')
'X' = Don't Care. Can be '0' or '1'
'-' = ignored
```

Output Field

```
'0' = Low
'L' = Low
'1' = High
'H' = High
'X' = unknown
'Z' = High Impedance
'-' = Not Specified
```

For a three-state-enabled pin (PAD), there should be at least one 'Z' state specified in the output field. Since the PAD pin appears at both the input and the output field, for any logic value row (vector), there should be a '-' at one of the field. This is required since a bidirectional signal should not be both input and output in the same vector.

Example

When the enable pin is set (PAD is output enabled), the following are valid (PAD @ PAD) logic values:

```
PAD @ PAD
===========
 -  @  0    <-- enable
 -  @  1    <-- enable
```

When the enable pin is unset (PAD is input), the following are valid (PAD @ PAD) logic values:

```
PAD @ PAD
===========
 0  @  -    <-- disable
 1  @  -    <-- disable
 Z  @  -    <-- disable (Liberate converts to "- @ Z")
 -  @  X    <-- disable
 X  @  -    <-- disable
```

Note: For any 'Z' found at the input of a bi-directional pin, if the output is '-', Liberate changes the output '-' to 'Z'.

Template Generated

The following Liberate commands are written to the generated template file: define_cell, define_pin_load, define_leakage and define_arc.

The define_pin_load command is added when there is a three-state enable pin in the truth table and at least one of PULLUP_RES, PULLDOWN_RES, PULLUP_VOLT, or SERIES_RES commands are specified.

See the **read_truth_table** command for an example.

Truth Table Flow

Use the Tcl command read_truth_table to read in truth table file(s). All loaded cell information is attached to the library specified in this command.

Use the Tcl command write_template with the -truth_table option to write out a template using the truth table data that was read in with the read_truth_table command.

After the template has been generated, it should be inspected and modified as required to suit the circuit.

Example:

To read in a truth table file (io_cells.tab), which contains several truth tables, and generate a Liberate template with 5-by-6 delay and power indexes.

```
read_truth_table io_cells.table
write_template -auto_index \
    -index_delay 5x6 \
    -truth_table template_io_cells
```

B

Constraint-related Delay and Clock Path Measurement

It is possible to have Liberate measure the clock and data paths for a given constraint arc. These path measurements can be reported in the library database (LDB) as described below, or can be used to compute the setup/hold time in lieu of the standard bisection search algorithm (see `define_arc -metric path_delta`). The measurements must be completely specified by the user. A path delay can be measured from a define_arc **pin** to **pin_probe** and from **related_pin** to **related_probe**. The following changes were made to support this functionality:

Liberate will perform an additional "final simulation" (last iteration + 1) with the requested clock and data path measurements possibly with added margin (back-off).

If set_constraint **-margin** is applied before char_library, the specified margin will be used as a back-off adjustment for the final simulation.

The **define_arc** command has the following options: **pin_probe**, **pin_probe_dir**, **pin_probe_threshold**, and **related_probe_threshold**. The threshold parameters accept values between 0 and 1 and default to a value of 0.5.

The following attributes will be written into the LDB: **altos_pin_probe_rise**, **altos_pin_probe_fall**, **altos_related_probe_rise**, and **altos_related_probe_fall**.

Example:

```
define_arc \
    -type setup \
    -metric path_delta
    -pin {D} -pin_dir -R \
    -related_pin {CK} -related_pin_dir R \
    -pin_probe_threshold 0.4 -related_probe_threshold 0.4 \
        -pin_probe {INT_1} \
    -related_probe {INT_2} \
DFFX1
```

The LDB will have:

```
      timing () {
        altos_pin_probe_fall : "0.121926 0.125029 0.126677 0.131257 0.124245
    0.13021 0.133014 0.133937 0.134979 0.12743 0.141288 0.143912 0.14479 0.142544
    0.134338 0.222769 0.225745 0.227327 0.223372 0.214846 0.299318 0.301962 0.30321
    0.295557 0.287447 ";

        altos_pin_probe_rise : "0.0133118 0.0131942 0.0135056 0.0166397
    0.0189255 0.00918533 0.00933199 0.00917782 0.0123258 0.0152624 0.00113986
    0.00152897 0.00089008 0.00424648 0.00767286 -0.088543 -0.0885724 -0.0886433 -
    0.0878743 -0.0834282 -0.211588 -0.211584 -0.211563 -0.211518 -0.208395 ";

        altos_related_probe_fall : "0.0451655 0.05435 0.059468 0.0610231
    0.0395994 0.044886 0.0541113 0.0594423 0.0620969 0.0419871 0.0448538 0.0541429
    0.0596396 0.0636134 0.0446303 0.0439837 0.0530142 0.0582266 0.061947 0.0438122
    0.0437301 0.0526872 0.0577312 0.0607178 0.042053 ";

        altos_related_probe_rise : "0.0419939 0.0505755 0.0553957 0.0564752
    0.036051 0.0422824 0.0509746 0.0557763 0.0572595 0.0367968 0.0422869 0.0510556
    0.0560625 0.0580703 0.0379491 0.0427997 0.0520553 0.0574491 0.0617163
    0.0433321 0.0426693 0.0519382 0.0572984 0.0616856 0.0444859 ";

        related_pin : "CK";
        timing_type : setup_rising;
        rise_constraint (constraint_template_5x5) {
          index_1 ("0.021898, 0.0788762, 0.135854, 0.591678, 1.16146");
          index_2 ("0.021898, 0.0788762, 0.135854, 0.591678, 1.16146");
          values ( \
            "0.0361599, 0.0288945, 0.0275717, 0.0445843, 0.0765158", \
            "0.0402818, 0.0325445, 0.0305551, 0.0464705, 0.0783839", \
            "0.0438295, 0.0357188, 0.0333824, 0.0494603, 0.0821265", \
            "0.0489605, 0.0411358, 0.0384973, 0.0587414, 0.0995835", \
            "0.0363275, 0.0290285, 0.027267, 0.0504022, 0.0963014" \
          );
        }
        fall_constraint (constraint_template_5x5) {
          index_1 ("0.021898, 0.0788762, 0.135854, 0.591678, 1.16146");
          index_2 ("0.021898, 0.0788762, 0.135854, 0.591678, 1.16146");
          values ( \
            "0.0269794, 0.0171525, 0.00999429, -0.000490503, 0.0155645", \
            "0.0338654, 0.0240763, 0.016751, 0.00486099, 0.0201764", \
            "0.0438901, 0.0341262, 0.0267148, 0.0132729, 0.0277857", \
            "0.118124, 0.108111, 0.100687, 0.082254, 0.0935043", \
            "0.192684, 0.182188, 0.174187, 0.149656, 0.156948" \
          );
        }
      }
```

C

External Simulator Options and Settings

Spectre

Spectre reads options from various locations including the `.options` statement, the `deck header`, the `.tran` statement, and the command line. *(*See the *Virtuoso Spectre Circuit Simulator Reference, Command Options* section for a full list of options.*)*

Passing options to Spectre requires setting the appropriate Liberate parameters to ensure that options appear in the expected location:

	Spice compatible	Native Spectre options	Transient control	Command line
Liberate parameters	extsim_option extsim_leakage_option	extsim_deck_header	extsim_tran_append	extsim_cmd_option
Spectre options	gmin method rabsshort save	reltol	errpreset lteratio	-64 +liberate +aps -cmiconfig +lorder +lqt +modellib +mt -mt +spice

Tip

The following is a series of examples showing how various options can be set to support different configurations or achieve different results.

General Spectre Settings for Accuracy and Performance

These represent a baseline set of options for Spectre when used with Liberate. The settings are accurate and recommended for all geometries.

```
set_var extsim_cmd_option       "+spice +lorder MMSIM:PRODUCT"
set_var extsim_deck_header      "simulator \
                                lang=spectre\nOpt1 options reltol=1e-4\nsimulator
                                \
                                lang=spice"
set_var extsim_leakage_option   "method=gear gmin=1e-15 redefinedparams=ignore
rabsshort=1m"
set_var extsim_option           "method=gear gmin=1e-15 redefinedparams=ignore
rabsshort=1m"
set_var extsim_tran_append      "lteratio=10"
set_var extsim_reuse_ic         3
```

Note: Liberate Spectre Aging flow was supported by Liberate 13.1 ISR1 along with MMSIM12.1 ISR16 or later versions. It is recommended to use the latest production version of Liberate along with the currently supported version of MMSIM (see `${ALTOSHOME}/README`).

Spectre Kernel Interface (SKI)

To enable SKI (the high-performance API to Spectre), use the following settings. The recommended version of MMSIM can be found in the `${ALTOSHOME}/README` file.

Note: These settings may change over time.

```
# Enable SKI
set_var ski_enable 1
set_var ski_alter_mode 3
# Clean up semaphores
set_var ski_clean_mode 2
# Enable "spectre +modellib" flow.  Use extsim_model_include with a comment as the
fist line.
set_var ski_enable_modellib 1
# Set SKI mdlthreshold to exact for best accuracy.
set_var ski_mdlthreshold_exact 1
```

```
# Set SKI power subtraction to match Spectre flow.
set_var ski_power_subtract_output_load_match_extsim 1
# Use the original name in the simulation.
set_var extsim_use_node_name 1
# Do not save simulation decks with SKI as the disk storage requirement is very
high.
set_var extsim_save_passed none
# Write SKI temporary files in an NFS partition in the working directory instead
of /tmp
set env(TMPDIR) "$env(PWD)/ski_temp"
```

Important

To use SKI, changes must be made to your `altos_init` file. An example is provided in the <release>/examples/ directory. We strongly recommend you use this altos_init as-is. Users who are familiar with modifying the altos_init file may merge their existing file with the example provided. The new altos_init should be placed in one of the following locations, in order of priority:

- run directory
- $HOME
- $ALTOSHOME/etc/

With SKI's current implementation, Liberate communicates with Spectre via shared memory and semaphores. One caution is that if a Liberate job is killed via bkill or qdel, shared memory may not be cleaned up properly. The altos_init script that is provided will perform this clean up automatically.

Note: When `extsim_model_include` is set, Liberate will check the format of the file and if compatible (that is, first line must not be a legal SPICE command), it will automatically enable the Spectre +modellib option. When using a full EMI flow, this can increase performance by ~30%.

Correlation between stand-alone Spectre and SKI

To achieve correlation between Spectre stand-alone, SKI, and Alspice add the following options. These are recommended as they will ensure consistent methodologies are used between the different simulations.

```
# Standardize transient window for Alspice/SKI and extsim decks
set_var power_sim_estimate_duration 1
set_var power_tend_match_tran 1
set_var tran_tend_estimation_mode 1
```

```
# Match SKI methodologies to Spectre
set_var ski_alter_mode 1
set_var ski_mdlthreshold_exact 1
set_var ski_power_subtract_output_load_match_extsim 1
# Match Alspice methodologies to Spectre
set_var alspice_power_subtract_output_load_match_extsim 1
```

LXI Support

LXI is an improved API that provides multiple improvements on the features of the default Spectre Kernel Interface (SKI). LXI increases measurement accuracy, provides additional debugging capabilities, and improves the accuracy in some corner cases.

To use LXI,

1. Set ski_meas_mode to `1`.
2. Add `+preset=ax -preset_override` to the extsim_cmd_option parameter to run Spectre X with LXI. To run Spectre APS, you do not require any additional command line options to be specified on the Spectre command line.

   Note: To use Spectre X with LXI, also ensure that `+liberate` is set on the Spectre command line.

If less accuracy is acceptable, additionally set the ski_mdlthreshold_exact parameter to `0` that ensures use of interpolated mode for Spectre Circuit Simulator Measurement Description Language (MDL) threshold.

Special Licensing

Cadence offers a characterization-only license for Spectre to increase cost-performance of library characterization known as the `spectre_char_opt` license. It is recommended to use this license if available.

Newer versions of Liberate automatically disables this feature if no licenses are available from the license server. Older versions would wait until the license server timed-out before switching to a different Spectre license, resulting in a delay at the start of every simulation. If no `spectre_char_opt` licenses are included, this feature may be disabled by setting the following parameter:

```
set_var spectre_use_char_opt_license 0
```

ECSM Accuracy and Correlation Settings for 20nm and Below

Use these settings to produce:

- Best-practices ECSM Libraries
- Production ECSM libraries for 20nm and below

Driver Cell

If the library is to be used only in an ECSM flow, then an active driver (or series of active drivers) has shown to provide the best accuracy. If the library is to be used in a mixed CCS/ ECSM flow, then studies need to be done by the end-user to determine which driver yields the best overall results.

Recommended Liberate Settings for ECSM

```
# Set ECSM modeling parameters for N-piece
set_var ecsm_version 2.1
set_var ecsm_cap_mode 1

# Receiver capacitance thresholds need to be balanced with respect to rise/fall
thresholds
set_receiver_cap_thresholds \
        -rise [list 0.1 0.3 0.5 0.6 0.7 0.8 0.9 0.9999] \
        -fall [list 0.9 0.7 0.5 0.4 0.3 0.2 0.1 0.0001]

# For ECSM waveform thresholds, the range is extended from 2% to 98%
define_template -type ecsm -index_1 {0.02 0.05 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8
0.9 0.95 0.98} ecsm_13

# Change the measured voltage range from fixed to dynamic.
set_var ecsm_measure_output_range 1

# Disable transition tables for tristate disable arcs
set_var tristate_disable_transition 1

# Update the ECSM-N model
set_var ecsm_arctype_enable 0
set_var ecsmn_loadcap_mode 1
```

```
set_var ecsmn_mode 1
```

See ecsm_measure_output_range for important considerations.

CCS Accuracy and Correlation Settings

Use these settings to produce:

- Best-practices CCS Libraries
- Production CCS libraries

Driver Waveform

The 2013.12 CCS Characterization Guidelines recommend using the released version of the CCS predriver waveform.

Recommended Liberate Settings for CCS

```
## Driver Waveform Settings
set_var predriver_waveform 2

## CCS Timing Settings
set_var ccs_voltage_tail_trim_tol 0.981
```

Note: Recommended simulator settings can change periodically. Therefore, Cadence recommends that all simulation settings used should be reviewed and tested before applying them in a production environment. Contact Cadence Customer Support for assistance.

D

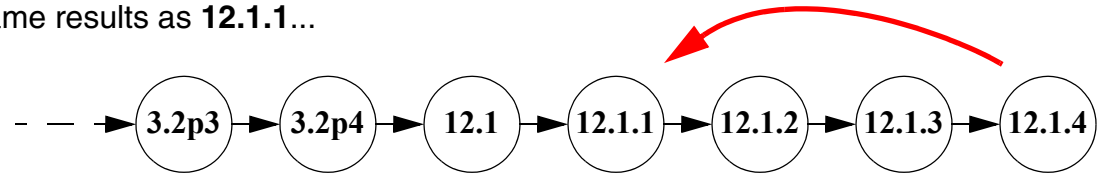
Qualifying and Migrating Between Versions

This appendix discusses the parameters that allow you to run your current version of Liberate and get the same results as an earlier version.

The parameters are presented as a Tcl script (`set_var` commands), and are organized to show which ones are needed to step back to a particular version. To use, include in your script all the `set_var` commands that go back to the version you need. (See example.)

Note: Default settings might be changed to comply with updated specifications, requirements from downstream tools, or to correct significant issues. Backward-compatible settings are also provided to make migration easier, but we strongly recommend using the new defaults after updating to the new version.

Example:
You have **12.1.4**, and you want the same results as **12.1.1**...



Use these set_var commands in your script.

```
# 12.1.4 to 12.1.3
set_var constraint_bisection_mode 1
set_var constraint_search_bound_estimation_mode 1
...

# 12.1.3 to 12.1.2
set_var
extsim_model_include_mode 1
...

# 12.1.2 to 12.1.1
set_var
ccsp_segmentation_effort 0
set_var
```

```
# 12.1.4 to 12.1.3
set_var constraint_bisection_mode 1
set_var constraint_search_bound_estimation_mode 1
set_var def_arc_constraint_use_arc_when true
set_var default_power_subtract_hidden_mode 1
set_var variety_netlist_mode 0

# 12.1.3 to 12.1.2.3
set_var extsim_model_include_mode 1
set_var mx_include_pull_driver false

# 12.1.2.3 to 12.1.2.2
# 12.1.2.2 to 12.1.2.1
# 12.1.2.1 to 12.1.2
# 12.1.2 to 12.1.1
set_var ccsp_segmentation_effort 0
set_var init_constraint_period_binning_mode 0
set_var variation_random_tran_thresh 0.5

# 12.1.1 to 12.1
set_var ccs_segmentation_effort 0x1
set_var ccsn_compatibility_multi_corners 0
set_var ccsn_dc_estimate_mode 0
set_var mx_greybox_constraint_method 0
set_var ski_elbr_mode 0

# 12.1 to 3.2p4_lcs
set_var bundle_log_filename ""
set_var ccs_cap_hidden_pin 1
set_var ecsm_cap_mode 0
set_var mx_leakage_check_time false
set_var mx_mpw_allow_same_probe_on_both_rise_and_fall_clock_tree true
set_var syscall_mode 0

# 3.2p4_lcs to 3.2p3
set_var aocv_derate_fit 1
set_var aocv_derate_mode defaultAVG
set_var ccsn_allow_multiple_input_switching 0
set_var ccsn_io_handle_tristate 1
set_var delay_glitch_detection_thresh 0.001
set_var read_library_expand_bundles false
```

```
set_var spice_param_eval_mode 1
set_var syscall_mode 2

# 3.2p3 to 3.2p2
# 3.2p2 to 3.2p1_lcs
set_var mx_postprocess_clock_probes true

# 3.2p1_lcs to 3.2
set_var mx_fastsim_reuse false
set_var mx_leakage_check_window 1
set_var mx_margin_report 0
set_var mx_power_assign " clock "
set_var mx_verbose false

# 3.2 to 3.1p4
set_var constraint_output_load "min"
set_var mx_rcdb_use_fastsim_deck false
set_var use_altos_default_group true

# 3.1p4 to 3.1p3
set_var constraint_max_risefall false

# 3.1p3 to 3.1p2
set_var driver_cell_acc_mode 0
set_var leakage_fix_ccsp true
set_var mx_hold_comb " none "
set_var mx_seq_probing "state output internal"
set_var mx_setup_comb " none "

# 3.1p2 to 3.1p1
set_var mx_whitebox_monitor_memcore false
set_var tryAsFloat 0

# 3.1p1 to 3.1
set_var aocv_nominal_swap_mode 0
set_var ccs_cap_use_input_transition_tristate 0
set_var mx_dynamic_include_full_core false
set_var mx_full_rail_tol 1e-2
set_var rc_compatibility_mode_3_0 true
```

```
# 3.1 to 3.0p3
set_var alspice_diode false
set_var ccsn_arc_channel_check 0
set_var mx_fullsim_measurement false
set_var mx_mpw_mode -1
set_var tcl_new_source false
set_var variation_ecsm_cap_input_pin false

# 3.0p3 to 3.0p2
# 3.0p2 to 3.0p1
# 3.0p1 to 3.0
set_var switch_cell_powerdown_function 1

# 3.0 to 2.5p2a
set_var char_mos_term_cap false
set_var combo_period 10e-9
set_var constraintCombinatorial 1
set_var extsim_sanitize_param_name false
set_var init_num_period 0
set_var library_copyright 0
set_var library_revision 0
set_var lic_queue_timeout -1
set_var miller_cap_factor 0.0
set_var mx_allow_blobs true
set_var mx_dpartition_inactive_tie "all"
set_var mx_false_probe_keep " none "
set_var mx_ring_max_xtr_cnt 1000000
set_var mx_ring_model_fold true
set_var psp_model_enable true
set_var switch_cell_powerdown_function 0

# 2.5p2a to 2.5p2a
# 2.5p2a to 2.5p1
set_var ccs_segmentation_effort 1

# 2.5p1 to 2.5
# 2.5 to 2.4p4
set_var ccs_init_voltage_comp_thresh -1
set_var ccs_segmentation_effort 0
set_var conditional_leakage false
set_var mx_char_virtual_as_rail " none "
```

```
# 2.4p4 to 2.4p3
# 2.4p3 to 2.4p2
# 2.4p2 to 2.4p1
# 2.4p1 to 2.4p1
# 2.4p1 to 2.4
# 2.4 to 2.3p2
set_var ccs_base_curve_share_mode 0
set_var ccsp_cross_zero_compact true
set_var ccsp_rel_tol 0.05
set_var ccsp_tail_tol 0.01
set_var compact_ccs_va_digits 10000
set_var compact_ccs_va_reltol 0
set_var mx_seq_probing_max_in 10

# 2.3p2 to 2.3p1
# 2.3p1 to 2.3
set_var ccsn_pin_stage_merge_mode 1

# 2.3 to 2.2p2
set_var ccs_base_curve_points 15
set_var ccsn_pin_stage_merge_mode 0
set_var mx_hold_comb 0
set_var mx_hold_seq 1
set_var mx_seq_probing_check_function true
set_var mx_setup_comb 0
set_var mx_setup_seq 1

# 2.2p2 to 2.2p1
set_var ccsp_current_using_sum_abs_reltol false
set_var opt_max_iter 200
set_var opt_strategy 1

# 2.2p1 to 2.2
set_var accurate_ccs_variation 0
set_var ccs_rel_tol 0.02
set_var mismatch_threshold2 1.0
set_var non_seq_pin_swap 1
set_var opt_max_iter 1000
```

```
# 2.2 to 2.1p1
set_var mx_hold_comb true
set_var mx_setup_comb true

# 2.1p1 to 2.1
set_var ccsn_pin_stage_lshift 0
set_var constraint_search_time_reltol 0.1
set_var ibis_interpolate 1

# 2.1 to 2.0p3
set_var ibis_iv_step 0.15
set_var mx_whitebox_active_coupling_threshold 1e-13
set_var mx_whitebox_ring_in_depth 2
set_var mx_whitebox_ring_out_depth 2

# 2.0p3 to 2.0p2
set_var default_method 0
set_var mx_whitebox_active_sidebranch true
set_var server_timeout 1209600

# 2.0p2 to 2.0p1
set_var mpw_input_threshold 0.5

# 2.0p1 to 2.0
# 2.0 to 1.4p3
# 1.4p3 to 1.4p2
# 1.4p2 to 1.4p1
set_var mismatch_count1 -1
set_var res_tol_ratio 0.02
set_var unidirectional_tristate false

# 1.4p1 to 1.4
# 1.4 to 1.3p3
# 1.3p3 to 1.3p2
set_var binning_detail 2
set_var ccsn_allow_static 1

# 1.3p2 to 1.3p1.2
# 1.3p1.2 to 1.3p1.1
# 1.3p1.1 to 1.3p1
# 1.3p1 to 1.3
```

```
set_var max_bdd_size 10000

# 1.3 to 1.2p3
set_var constraint_clock_gater false

# 1.2p3 to 1.2p2
set_var mismatch_count1 5
set_var mismatch_threshold1 0.5

# 1.2p2 to 1.2p1
set_var def_arc_drive_side_bidi false
set_var extsim_check true
set_var non_linear_variation 0
set_var predriver_waveform_npts 18
set_var spectrespp true
set_var vector_limit 16

# 1.2p1 to 1.2
set_var ccs_min_pts 5
set_var floating_node_check false
set_var predriver_waveform_npts 0

# 1.2 to 1.1p9
set_var floating_node_check true
set_var immunity_search_voltage_abstol 2e-3
set_var predriver_waveform_npts 18
set_var rc_parviews false

# 1.1p9 to 1.1p8
# 1.1p8 to 1.1p7
set_var predriver_waveform_npts 0

# 1.1p7 to 1.1p6
set_var floating_node_check false

# 1.1p6 to 1.1p5
set_var floating_node_check true

# 1.1p5 to 1.1p4
set_var arc_partition 10
```

E

Using Ascava Distillation Tool for Ultra Compaction of Liberate Files

Liberate is compatible with Ascava Distillation technology that produces compression of Liberty files up to twice of what the `tar` or `gzip` utility produces, and in a shorter run time. The LIBERATE 19.2 release includes a free trial of Ascava Distillation software.

Ascava distillation takes a file or directory of files and creates a single archive file (similar to `tar -zcf`). It can compress a single file or a directory of files (recommended) into a single archive. The more similar the files are in the directory, the better the distillation, and the higher the compression. Therefore, it is recommended to ultra-compress a directory full of Liberty `.lib` files from the same PVT because these `.lib` files will have a lot of redundancy, resulting in higher compression.

Ascava commands are located in the `$ALTOSHOME/bin` directory and are used as described in this appendix.

To distill (ultra compress), use the following command:

```
distill -a <archive_file>.dist [-e|--erase]  <path_to_directory_of_lib_files>
```

The `-e` argument is optional. It deletes the original archive after the compression is successful.

To undistill (ultra de-compress), use the following command:

```
undistill -a <archive_file>.dist -o|--output <unarchive_target_directory>
```

To list (print) the contents of a `.dist` archive (similar to `tar -t`), you can use the following command:

```
distill -p|--print -a <archive>.dist
```

OR

```
undistill -p|--print -a <archive>.dist
```

Using Ascava Distillation Tool in Liberate

In Liberate, you can add the distillation commands directly into your Tcl run script. However, ensure that this ultra compression command only runs on the Liberate server. You can do this as shown in the following example flow:

```
char_library ...
write_library ... -filename $rundir/lib/library.nldm.lib
write_library -ccs ... -filename $rundir/lib/library.ccs.lib
write_library -ccsn ... -filename $rundir/lib/library.ccsn.lib
if { [llength [packet_slave_cells]] == 0 } {
    exec distill -a $rundir/lib.dist $rundir/lib
}
```

The `if` statement above ensures that this command runs only on the Liberate serverserver process and not on all the worker client jobs.

Enabling Ascava Distillation Tool Trial License

Before Ascava tools can be used, you must activate the limited time trial license. To do this, use the following command:

```
$ALTOSHOME/bin/distill --eval-activation
```

This step provides you with a trial license token that can be set using an environment variable as shown below:

```
setenv ASCAVA_EVAL_TOKEN xxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Once this environment variable is set, you are ready to use the Ascava Distillation tool.

F

Deprecated and Backward Compatibility Commands and Parameters

This appendix lists all the commands and parameters that are deprecated, or included only for backward compatibility. We would like to discourage users from relying on these commands and parameters. You are instead encouraged to find alternate commands/ parameters and settings to achieve the best results from Liberate.

This appendix covers the following topics:

Deprecated Commands

The commands covered in this section have been phased out, and have been replaced by a new command, parameter, or behavior of the tool.

a...	
add_lib_attribute	add_pin_attribute
add_cell_attribute	
d...	
define_arc -constraint (only one option has been deprecated)	define_em
r...	
remove_pin_attribute	
s...	

set_ccs_retry_thresholds	set_em_imax
set_client	
w...	
write_template -input_supply_pin (only one option has been deprecated)	

add_lib_attribute

`{attributes}` Specifies the simple or complex attribute(s) to add to a library.

add_cell_attribute

`{cells} {attributes}`

Specifies the simple or complex attribute(s) to add to cell(s) in a library.

add_pin_attribute

`{cells} {pins} {attributes}`

Specify the simple or complex attribute(s) to add to the pin(s) of cell(s).

define_arc -constraint

In the `define_arc` command, the `-constraint` argument has been deprecated. Use the `-logic_constraint` argument instead.

define_em

Defines the cells and resistors and ports that will be characterized for electromigration models.

Options

`-mode`	Model mode, can be {ac}, {dc} or {ac dc}.
`-resistor {R, Port}`	Specifies the resistor names and output port names for EM acquisition.
`{cellnames}`	List of cells for EM characterization.

The resistor name can be a wildcard (`"*"`) and supports a trailing wildcard (`"name*"`). The `ac` and `dc` modes are used in the proprietary calculation of `max_toggle_rate` library values. In the `dc` mode, the internal currents are summed for both rising and falling toggles in the same simulation. In `ac` mode, each edge is modeled separately. In Chapter 7, "Performing Characterization using Liberate," the Electromigration Models section describes these modes in detail. This command supports wildcard characters.

This command must be used before `char_library`.

Example

```
define_em -mode {ac dc} -resistor {R1  Y} INVX1
define_em -mode {dc}    -resistor {R2   } {INVX1 NANDX1}
```

remove_pin_attribute

Removes the automatically-generated pin-level attributes from the output Liberty file.

Options

`{cells}`	List of cells.
`{pins}`	List of pins.
`{attribute list}`	List of attributes.

Pin-level attributes can be provided in the write_library `-user_data` file or automatically added by Liberate. To remove the pin-level attributes that are supplied in the `user_data` file, modify the `user_data` file.

Note: This command supports wildcards.

Examples

```
# Remove input_signal_level from pins Q and CK on cell DFF
remove_pin_attribute {DFF} {Q CK} {input_signal_level}

# Same as above except all pins of all cells
remove_pin_attribute {*} {*} {input_signal_level}
```

set_ccs_retry_thresholds

Allows users to override the default behavior of Liberate with respect to voltage thresholds checked during CCS-T characterization.

Options

`-rise {`*`list`*`}`	List of voltage thresholds specified as a percentage in decimal. (for example, `.5` = `50%`)
`-fall {`*`list`*`}`	List of voltage thresholds specified as a percentage in decimal. (for example, `.5` = `50%`)

The default is for Liberate to check the following:

- `rise`: `measure_slew_lower_rise`, `delay_out_rise`, `measure_slew_upper_rise`
- `fall`: `measure_slew_upper_fall`, `delay_out_fall`, `measure_slew_lower_fall`

This command has no effect if `ccs_retry_mode=0`. If `ccs_retry_mode` is set to `1` or `2`, Liberate checks the reconstructed voltage waveform derived from the output current waveform at these thresholds against the `ccs_abs_tol` and `ccs_rel_tol` criteria. If the tolerance criteria are not met, the simulation will be retried with more accurate options.

This command must be used before `char_library`.

Example

```
# Specify 20%, 30%, %50, 70%, 80%
# for both rising and falling arcs
set_ccs_retry_thresholds \
    -rise [list 0.2 0.3 0.5 0.7 0.8] \
    -fall [list 0.2 0.3 0.5 0.7 0.8]
```

set_client

Options

`-n <number_of_clients>`

Specifies the number of clients to use.

If the `-n` argument is specified, Liberate submits the jobs to the specified number of clients with the specified queue name. The directory name (`-dir`) can be created using some or all of the following values:

- `%N` (denotes the client number)
- `%U` (denotes the user name)
- `%S` (denotes the server name)
- `%P` (the Liberate server process ID)

Important

The `-n` argument explained above is deprecated. Use the packet mode instead of the `set_client` mode to use a queuing system. For more details on distributed parallel processing, see Chapter 3, “Parallel Processing.”

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

Defines a directory on the client machine to use as a temporary workspace for simulation jobs performed on that machine. Liberate creates the directory if it does not exist. (REQUIRED)

`<Queue_name>` Name of the batch queue.

These can also be used to create unique scratch directories for each individual Liberate characterization run.

Note: When the `-n` argument is used, all file names within the Tcl file must be full path names.

It is also recommended that Liberate is executed with a Tcl file specified using a full path name.

The `-n` argument is only supported when an external job-queuing system is used. When Liberate is managing the clients using its built-in queuing system and the `rsh_cmd` parameter is set to `rsh` or `ssh`, the `-n` argument should not be used. Alternatively, Liberate can 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.

set_em_imax

Specifies the maximum current value and the resistors or output pins to measure. The `-resistor` and `-cells` options both accept a wildcard (`"*"`). For more information, see Electromigration Models.

Options

-cells {*list*}	List of cell names.
-resistor {*resistor* | *output_names*}	
	List of resistor names or output pins.
-type <"avg" | "rms">	
	Average or RMS.
<*value*>	IMAX value.

This command must be used before `write_library`.

Example

```
# Set the man current for resistor R1 to 5mA
set_em_imax -type avg \
    -resistor { R1 } \
    -cells { inv and2 }
    5e-3
```

write_template -input_supply_pin

In the `write_template` command, the `-input_supply_pin` option has been deprecated.

Deprecated Parameters

The parameters covered in this section have been phased out, and have been replaced by a new command, parameter, or behavior of the tool.

b...	
bundle_count	bundle_when
c...	
ccsn_allow_probe_driven_by_feedback	cell_use_both_ff_latch_groups
ccsn_overlap_ccr_include_mode	conditional_immunity
ccsp_report_segmentation	
d...	
debug_flow	default_power
default_capacitance	default_timing
default_group_method	default_unateness
default_leakage	
e...	
em_period	extsim_model_include_leakage
extsim_interactive	
l...	
leakage_accuracy_mode	
m...	
mpw_delay_use_active_edge	
p...	
packet_arc_licensing_mode	packet_arc_require_spectre_char_opt
r...	
rc_sort_mode	

bundle_count

<value>	Specifies a specific number of bundles. Default: follow bundle_mem_limit.

Bundle mode is enabled when bundle_mem_limit is exceeded or when this parameter is set to a number of bundles. Use this option to override the bundle count that is automatically determined when bundle mode is enabled by bundle_mem_limit. For example, if there are 100 cells and bundle_count is set to 2, there will be 2 bundles with 50 cells in each bundle.

By default, the bundle mode is not enabled. This feature has been deprecated in favor of the packet mode.

Note: bundle_count is not compatible with either the cell-based or arc-based packet mode. If packet_clients>0, do not set bundle_count>0.

This parameter must be used before char_library.

bundle_when

<0 | 1>	Controls the mapping of bundles in a when condition. Default: 1
0	Use a reference bit from the bundle in the when condition. The bit is controlled by the -use_pin option of the define_bundle_pins command . For example: `when : "D1"`

1	Use the bundle name as specified in the define_bundle_pins command in the when condition.
	This setting assumes that all members of the bundles have the same state value:
	`Supported:`
	`  when "D"  == when " D2 &  D1"`
	`  when "!D" == when "!D1 & !D2"`
	`Not possible if using bundle name in the when:`
	`  " D1 * !D2"`
	`  "!D1 *  D2"`

Bundle pins that occur in `when` or `sdf_cond` can either use the bundle name, or a reference bit from the bundle. For information on defining bundles, see the define_bundle_pins command. In the example given below, assume a bundle being defined as follows:

```
define_bundle_pins -use_pin { D1 } MyCell  D  { D2 D1 }
```

This parameter must be used before the char_library command.

ccsn_allow_probe_driven_by_feedback

`< 0 | 1 >`

There could be a latch up condition in the Channel Connected Component/Region (CCC) that is driving the out probe (for example, through a passgate). This parameter checks if the out probe is connected to any such loop wire and tries to choose an alternate CCC output, if available. Default: `0`

- `0` Avoids using such probes for CCSN characterization.
- `1` For backward compatibility.

ccsn_overlap_ccr_include_mode

`< 0 | 1 >`

Updates the handling of CCB when the input is (also) connected to the output through a passgate thus providing an alternate path. Set this parameter to `1` for backward compatibility. Default: `0`

ccsp_report_segmentation

`<0 | 1 | 2>`

Checks that the CCSP waveform data in the library is similar in peak and area under the curve when compared to the raw simulation output waveform.
Default: `0`

- `0` Disables the check.

1 Generates a summary report in the log file. Below is an example of the report:

```
CCSP summary report:
    passing cells: 4/4
    failing cells: 0/4
    skipped cells: 0/4
```

2 Generates a comparison report in the output report file, `comp_ccsp.txt`, and a summary report in the log file.

This parameter must be set before the char_library command is run. However, its impact on the CCSP waveform data can be noticed after the write_library command is run.

Example

Below is a sample of the comparison report:

```
=== comp_ccsp.txt ===
*** BEGIN PINATOPINB COMPARISON ***


A1(R)->Z(R) ( VDD VSS )
sim_area       sim_peak      ccsp_area      ccsp_peak      diff_area     diff_peak
diff_area(%)  diff_peak(%)
 2.587536e-15   4.789411e-05   2.619237e-15   4.789411e-05   3.170190e-17
0.000000e+00       1.2252%       0.0000%
-3.118631e-15  -3.613516e-05  -3.087927e-15  -3.613516e-05   3.070389e-17
0.000000e+00      -0.9845%      -0.0000%
 4.099531e-14   1.678425e-04   4.088179e-14   1.678425e-04  -1.135194e-16
0.000000e+00      -0.2769%       0.0000%
-2.946961e-15  -2.320267e-05  -2.998395e-15  -2.320267e-05  -5.143417e-17
0.000000e+00       1.7453%      -0.0000%
... ...
-----------------------------------------------------------
|   type    | Entries | Avg Diff% |   Pass% | Outliers |
-----------------------------------------------------------
| ccsp area |      24 |     0.11% | 100.00% |        0 |
-----------------------------------------------------------
| ccsp peak |      24 |     0.00% | 100.00% |        0 |
-----------------------------------------------------------
*** END PINATOPINB COMPARISON ***
```

cell_use_both_ff_latch_groups

`< 0 | 1 | 2 >` Enables support for multiple *ff* and *latch* groups in the output library. Default: `1`

- `0` Liberate includes one of the latch groups in the output library if the `write_library` user_data file contains both *ff* and *latch* groups.

  Note: This setting is provided for backward compatibility.

- `1` Liberate enables the modeling of both the *ff* and *latch* groups, but keeps only one of each if there are multiple *ff* and *latch* groups in the user_data file.

- `2` Liberate enables the modeling of multiple *ff* and *latch* groups. This is the recommended setting if multiple *ff* and *latch* groups are needed to correctly model the cell function.

Some complex cells, such as retention DFFs use the simultaneous modeling of *ff* and *latch* groups in the Liberty file for correct function modeling. Use this parameter to enable support for modeling multiple function groups.

The *ff*, *latch*, *ff_bank*, and *latch_bank* must be provided with the `write_library` user_data file.

To model *ff_bank* and *latch_bank* groups, the `define_bus` and/or `define_bundle` commands must also be used. The *ff*, *latch*, *ff_bank*, and *latch_bank* groups can be modeled under a cell, as described below.

	ff/latch	ff_bank/ latch_bank
buses/bundles	yes	yes
no buses/bundles	yes	no

This parameter must be set before `write_library`.

conditional_immunity

`<0 | 1>` Controls characterization of conditional Noise Immunity Curve (NIC) data.
Default: `0`

`0`	Outputs the worst case NIC.
`1`	Request the output of unique NICs for each `when` condition **Note:** Using this value increases the run time.

This parameter must be used before the `char_library` command.

Example

```
# Turn on conditional states for noise immunity
set_var conditional_immunity 1
```

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.

Note: This parameter has been deprecated. Use of the select_index command is recommended instead of this parameter.

For variation characterization in Liberate Variety or in unified characterization flow of Liberate Trio, the minimum dimension of a template is 2x2.

This parameter must be specified before the first `define_template` command.

Example

```
set_var debug_flow 2x2
```

default_capacitance

`<min | avg | max>` Sets the desired selection criteria for the *capacitance* attribute. Use the minimum (`min`), average (`avg`), or maximum (`max`) rise/fall capacitance value as the default pin rise/fall capacitance. Default: `max`

Note: This parameter has been deprecated. Use of the set_default_group command is recommended instead of this parameter.

This parameter must be set before `char_library`.

default_group_method

`< 0 | 1 >` Specifies the method to use for creating default groups. Default: `1` (use the whole table with the worst bit)

Note: As this is a deprecated parameter, use of the set_default_group command instead is recommended.

- `0` The worst value **(**`min` or `max`) from all the relevant tables on a bitwise basis will be individually used for all data types. If `default_power` is set to `avg`, a point-to-point average value from all the power groups is used for the default power table.
- `1` Find the delay, power and cap table that has the worst value (reviewed bitwise) and to use that complete table, in its entirety, in the default group. This parameter does not apply to constraints. Constraint default groups are always selected on a bitwise basis. (Default)

This parameter must be set before `char_library`.

default_leakage

`< min | avg | max >` Use the minimum (`min`), average (`avg`), or maximum (`max`) leakage current value as the default leakage current. Default: `avg`

Note: This parameter has been deprecated. It is recommended that the set_default_group command be used instead of this parameter.

This parameter must be used before `write_library`**.**

default_power

`< off | min | avg | max >`

Either ignore (`off`), use the minimum (`min`), the average (`avg`) or the maximum (`max`) power as the value for the default power group. Default: `max`

Note: This parameter has been deprecated. It is recommended that the set_default_group command be used instead of this parameter.

This parameter must be used before `char_library` if it is to have any effect.

default_timing

`< off | min | max | force_off >`

Either ignore (`off` or `force_off`), use the `min` or `max` delay, transition and constraint values as the value for the default timing group. Default: `max`

Note: This parameter has been deprecated. It is recommended that the set_default_group command be used instead of this parameter.

This parameter must be used before `char_library`.

default_unateness

`< merge | separate >`

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

Note: This parameter has been deprecated. It is recommended that the set_default_group command be used instead of this parameter.

These parameters control the creation of default library group data for pin capacitance, leakage, power and timing. The default group data is created from taking the `max` or `min` of the values from the relevant conditional groups. The default timing or power groups can be omitted by setting the `default_timing` or `default_power` to `off`. Setting this parameter to `separate` will keep these groups distinct.

Note: If all *timing_sense* attributes are identical, then the original timing sense will remain unchanged during merging. If there are both *positive_unate* and *negative_unate* timing sense group data, then the merged timing sense will be *non_unate*.

This parameter must be used before `char_library`. Examples:

```
# Set defaults for the fast corner
set_var default_power min
set_var default_capacitance min

# Ignore default timing groups
set_var default_timingoff

# Disable merging for binate default groups
set_var default_unateness separate
```

By default Liberate will create a single default timing group by merging *positive_unate* and *negative_unate* default groups into a single *non_unate* timing group. Setting this parameter to `separate` will keep these groups distinct. This command typically has an impact on complex combination cells, such as adders.

em_period

`< value >`	Specifies the clock period (in seconds) at which to characterize electromigration (EM). The value specified must be a number in seconds. Default: `" "`

The `em_clock_freq` overrides `em_period`. It is recommended that only one of these two parameters is specified. For more information, see the Electromigration Models section in Chapter 7, "Performing Characterization using Liberate."

This parameter must be used before `char_library`.

Example

```
# Set the EM Clock Period to 50nS
set_var em_clock_freq  50e-9
```

extsim_interactive

`<0 | 1>`	Controls whether the external simulator should be operated in an interactive mode when the `char_library -extsim` command is set. Default: `0 (disabled)`
`0`	Corrects the settings of the previously mentioned commands if you have problems with hanging jobs in your environment.
`1`	Specifies to operate the external simulator in an interactive mode when it is enabled with `char_library -extsim`.

Running the external simulator in interactive mode reduces the external simulator start-up time and can be useful in network environments that are not configured for efficient cell characterization runs. One currently known issue is that using Eldo in interactive mode can leave orphaned processes that will need to be manually killed if the Liberate or Liberate Variety run exits abnormally.

Note: This functionality only supports Eldo 2008.06 or later. Liberate uses the interactive mode that starts Eldo only once upfront instead of starting Eldo for each simulation deck. You need to take care not to start more interactive jobs than available license tokens by using appropriate settings for `set_client`, `char_library -thread`, and simultaneous job submission. Failure to do this may result in hanging Liberate and Eldo jobs. If you have

problems with hanging jobs in your environment, correct the settings of the previously mentioned commands or disable `extsim_interactive` by setting it to `0`.

This parameter must be used before the `char_library` command.

Example

```
set_var extsim_interactive 1
```

extsim_model_include_leakage

`<"`*`model_file`*`">`	Specifies full path to model file. Default "" (none)

Set this to characterize leakage with a separate process model file. This will `.include` the leakage model into the leakage decks. (This will not affect the normal read_spice flow on the regular process models.) The leakage decks written out will not flatten the netlist.

This parameter must be used before the `char_library` command.

Example

```
set_var extsim_model_include_leakage "/home/myDir/model.inc"
```

leakage_accuracy_mode

`< 0 | 1 >`	Improves leakage accuracy. Default: 1	
	`0`	Liberate uses pre-processed netlist for leakage simulation.
	`1`	Liberate retains all R/C.

This parameter is relevant for netlists flattened by Liberate.

If the netlist is not flattened by Liberate (`extsim_model_include` and `define_leafcell` are specified and `extsim_flatten_netlist` is set to 0) then the netlist and model are included in the deck and leakage_accuracy_mode is essentially non-operational. This is the recommended flow for FinFET technologies.

This parameter must be used before `char_library`.

mpw_delay_use_active_edge

`< 0 | 1 >` Specifies which clock edge to use when characterizing MPW. Default: `1`

Set this parameter to enable the measurement of mpw related delay degradation from the active edge of the circuit clock instead of the leading edge of the clock. By default, when `mpw_criteria = 1` (delay degradation), Liberate will measure the delay degradation from the leading edge of the 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. The default and recommended setting for this parameter is `1`.

This parameter must be used before `char_library`.

packet_arc_licensing_mode

`< default | wait_for_clients >`

Controls license-handling for arc-based packet mode. Default: `default`

Note: This parameter has been deprecated and should not be used any longer.

`default` All Liberate clients will release their Liberate and Spectre licenses as soon as characterization has completed on that job. Recommended for installations with one server or where the load balancing software manages the tool license usage. (Default and Recommended)

`wait_for_clients`

The Liberate server performs block check-outs of `Liberate_client` and `Spectre_char_opt` licenses based on client license availability. If insufficient licenses are available, the server continues to query the license server for additional licenses.

The Arc-based Packet Mode provides flexible usage of the Liberate_client and SPICE licenses . The optimal method depends on the number of Liberate servers, Liberate clients, Spectre licenses, and demand from other users for the Spectre licenses. For flows not using Spectre, only the number of Liberate licenses needs to be considered.

This parameter must be used before `char_library`.

packet_arc_require_spectre_char_opt

`<0 | 1>` Defines how the arc packet flow handles Spectre_char_opt licenses.
Default: `0`

- `0` The `packet_client` continues even if it fails in checking out the required number of `Spectre_char_opt` licenses. However, each characterization task on this `packet_client` checks out `Spectre_char_opt` licenses when the task starts and checks them back in at the end of the task. If `packet_client` fails to check-out `Spectre_char_opt` licenses, it will let Spectre handle its licensing and check-out other licenses such as, `Virtuoso_Multi_mode_Simulation`, `Virtuoso_Spectre`, or `Virtuoso_Acceler_Parallel_sc`, as required. Therefore, excessive license check-out and check-in occurs if `Spectre_char_opt` licenses are insufficient. This is because each `packet_client` typically runs `10s` of tasks.

`1`	*(Recommended if enough* `Spectre_char_opt` *licenses are available.*) The `packet_client` waits until it checks out the required number of `Spectre_char_opt` licenses. The `packet_client` during its initialization phase checks out the required `Spectre_char_opt` licenses upfront according to the number of threads and simulation type. Simulations involving electromigration require ***two*** `Spectre_char_opt` licenses per CPU thread. However, other simulations require ***one*** `Spectre_char_opt` license per CPU thread. In this mode, Liberate checks out `Spectre_char_opt` licenses and does not attempt to check out other Spectre licenses such as, `Virtuoso_Multi_mode_Simulation`, `Virtuoso_Spectre`, or `Virtuoso_Acceler_Parallel_sc`.

This parameter must be used before the `char_library` command.

packet_rdb_mode

`<0 | 1>`	Enable RDB in parallel packets. Default: `0` (disabled).

The RDB is a recovery data base that saves raw SPICE data in a compact form. The RDB is used to store data for incremental or arc-based flows when using a parallel packet mode. If kept, recharacterizations will take much less time, as Liberate can skip a simulation if the results are already present.

0: Disables RDB in parallel packets. (Default)
1: Enables RDB in parallel packets.

This parameter will be automatically set to 1 when `packet_mode=arc`. The recommended setting is `0` in cell packet flow.

This parameter must be used before the `char_library` command.

parse_space_bang_is_comment

`<0 | 1>` Specifies whether to treat bang (!) preceded by a space as a comment.
Default: `0`

- `0` Parses the netlist and does not treat bang (!) preceded by a space as a comment.
- `1` Accepts bang preceded by a space as a comment when parsing the netlist.

This parameter must be used before the read_spice command.

rc_sort_mode

`< 0 | 1 >` Enables sorting of RC elements. Default: 1

This parameter lets you decide how to add parasitics to Alspice or external simulator decks to help prevent consistency issues when running a single cell versus an entire library. By default, Liberate enforces a consistent order for both single and multi-cell runs. If set to 0, Liberate restores the behavior of the 3.0p2 and prior releases where RC elements are never sorted.

Note: This parameter was deprecated with release 3.2p4 and will be ignored with a warning. The RC elements will always be sorted for consistency.

write_template_force_power

`<0 | 1>` Output power templates whenever there is characterized power data.
Default: `1`

The `write_template` command outputs a reference to a power template for every cell that has any power data. There are cells, such as, DECAP and Filler cells, that do not have any timing or internal power arcs, but that only have leakage. To be able to characterize these types of cells, Liberate must have a `define_cell -power` option referencing a power template (see define_template).

Set this parameter to `0` to get release v3.2p3 and prior behavior where `write_template` would not include a `-power` option in the `define_cell` command for these types of cells.

This parameter must be set before the write_template command is run.

Backward Compatibility Parameters

The parameters covered in this section invoke the older behavior of the tool. Cadence discourages using these parameters because many of the older algorithms have since then been corrected or improved.

c...	
■ capacitance_force_hidden	■ ccsn_model_unbuffered_output
■ ccsn_active_ccr_recognition_mode	■ ccsn_pin_unconditional
■ ccsn_allow_multi_switching_unate_groups	■ ccsn_prefer_two_sided_stages
■ ccsn_check_arc_level_reconvergence	■ ccsn_probe_mode
■ ccsn_check_char_values	■ ccsn_redundant_pin_stages
■ ccsn_check_valid_noise_prop	■ ccsn_skip_mod_vdata
■ ccsn_compatibility_mode	■ ccsn_unateness_based_merge_vecdata
■ ccsn_compatibility_multi_corners	■ ccsn_unbuffered_output_mode
■ ccsn_dc_estimate_mode	■ cell_leakage_power_legacy_mode
■ ccsn_dc_sweep_mode	■ char_mos_term_cap_ski_mode
■ ccsn_fanout_select_mode	■ conditional_expression_accuracy
■ ccsn_io_skip_channel_inputs	■ constraint_dependent_nonseqsetuphold
■ ccsn_io_skip_channel_input_netlisting	■ constraint_search_time_reltol_mode
■ ccsn_input_xfr_probe_mode	■ constraint_slew_degrade_nominal_check
■ ccsn_mega_mbit_hidden_mode	■ constraint_stim_precision_auto_adjust
■ ccsn_mega_mbit_skip_overlapping_vdata	
d...	
■ default_non_unate_rcvr_cap_adjust	■ def_arc_delay_metric_mode
■ default_power_subtract_hidden_mode	■ disable_current_measure_effort
■ default_timing_tristate_enable	■ driver_waveform_lib_mode

e...	
■ ecsmn_allow_duplicate_condition ■ ecsmp_invert_gnd_current ■ em_check_lib_data ■ em_check_table_data	■ em_unmatched_resistors_check ■ extsim_model_include_multi_vector_mode ■ extsim_save_driver
f...	
■ floating_node_consistency_check	
i...	
■ ignore_dummy_fets	
m...	
■ mega_compatibility_mode	■ model_operating_condition_mode
n...	
■ nonseq_as_recrem_char_mode	■ non_seq_probe_mode
p...	
■ parse_sfe_parser_mode ■ power_multi_vector_mode	■ power_subtract_leakage_tran_mode
r...	
■ reset_negative_power_mode	
s...	
■ set_pin_slew_threshold_mode ■ si_write_output_voltage_compatibility_mode ■ simultaneous_switch_from_cell_when	■ switch_cell_dc_partition_include_primary_input ■ switch_cell_infer_unateness_from_ldb
t...	
■ tristate_pin_cap_use_arc	
u...	
■ user_data_keep_simple_attr_quotes	
v...	
■ vec_data_sort_mode	■ voltage_map_ldb_char_mode

w...	
■ write_logic_function_async_mode	

capacitance_force_hidden

`< 0 | 1 >` Enables characterization of input capacitance for hidden power arcs. Default: `1`

- `0` In IO mode (see the -10 otpion of the char_library command), Liberate does not characterize input pin capacitance for hidden power arcs when the pin does not have any forward arcs. The following warning is printed:

  ```
  *Warning* (char_library): Capacitance acquisition is disabled for hidden arc on pin PD_EN.
  ```

- `1` Enables input capacitance characterization on hidden power arcs. No warning is printed since the input capacitance is characterized.

This parameter must be used before the `char_library` command.

ccsn_active_ccr_recognition_mode

`< 0 | 1 >` Specifies whether to do a more aggressive search for CCSN arcs. Default: `1`

If this parameter is set to `1`, Liberate will more aggressively search for CCSN arcs. It will search for the existence of CCSN paths from input to outputs, based on the actual vector being sensitized to remove inactive CCSN paths. The recommended setting is `1`.

This parameter must be used before the `char_library` command.

ccsn_allow_multi_switching_unate_groups

`<-1 | 0 | 1>` Controls characterization of positive unate CCBs that have multiple inputs switching simultaneously.
Default: `-1`

-1	Always enabled (=1) for the input_ccb/output_ccb format (ccsn_use_io_ccb_format=1) and disabled (=0) for the ccsn_first_stage/ccsn_last_stage format (ccsn_use_io_ccb_format=0).
0	Never characterize the CCBs.
1	Always characterize the CCBs.

This parameter must be used before the char_library command.

ccsn_check_arc_level_reconvergence

< 0 | 1 >

Controls whether Liberate should check for reconvergence between single-stage Channel Connected Block (CCB) and multi-stage CCB in determining the pin-level or arc-level CCSN groups.
Default and recommended: 1

0	Do not check for reconvergence. This value is available for backward compatibility with only the LIBERATE 16.1 ISR1 release.
1	Checks for reconvergence.

This parameter must be set before the char_library command.

ccsn_check_char_values

`<0 | 1 | 2>` Checks that the range of values is correct for the `output_voltage_*` and `propagated_noise_*` tables in the CCSN groups.
Default and recommended: `1`

- `0` Do not check the correctness of the range of values. This value is available for backward compatibility with only the LIBERATE 16.1 ISR1 release.
- `1` Checks only the pull-up and pull-down CCSN groups.
- `2` Checks all CCSN groups.

This parameter must be set before the char_library command.

ccsn_check_valid_noise_prop

`<0 | 1 | 2 | 3>` Checks if a CCB can propagate noise during vector generation prior to characterization.
Default (and Recommended): `3`

- `0` Disallows filtering of CCBs that cannot propagate noise prior to characterization. The characterization and modeling log files contain warnings about the arcs that fail.
- `1` Checks for a tristated CCB output for rise or fall transitions. If found, skips the CCB for noise analysis. This saves run time and prevents log file warnings.
- `2` Use this setting for backward compatibility with LIBERATE 13.1 ISR4 and prior releases.
- `3` Performs the checks done when the parameter is set to `1`, and then characterizes the one-sided stages if found.

Under specific WHEN conditions, the output of some CCBs might change only in one direction and not allow the CCB output to switch in the opposite direction. Under these conditions, no noise propagation can be modeled and only a static noise group exists.

However, downstream tools do not handle static noise groups presently. Therefore, Liberate issues a warning and filters these out prior to model generation. The `ccsn_check_valid_noise_prop` parameter allows these CCBs to be removed during vector generation. This prevents them from being characterized for CCSN, saves run time, and removes the related warnings from the log file. The resulting CCSN Liberty model is identical, regardless of the setting of this parameter.

This parameter must be set before the char_library command.

ccsn_compatibility_mode

`< 0 | 1 | 2 >` Controls checks to more thoroughly analyze CCB topologies and electrical paths to identify and characterize CCSN stages at the timing and pin level. Default: `1`

- `0` Use standard level of CCB topology checking.
- `1` Use additional checks that correct cross-PVT compatibility issues. (Default and recommended if voltage/temperature scaling will be used on the resulting libraries.)
- `2` Use the `char_library -io` mode CCSN algorithm when the delay-based vector generation algorithm does not produce the expected results.

This parameter must be set before the `char_library` command.

ccsn_compatibility_multi_corners

`< 0 | 1 >` Effects CCSN structure in library. Default: `1`

- `0` Backward compatible with release `3.2p4_lcs` where different PVTs can have structure differences in CCSN data.
- `1` Maintain structure equivalence across PVTs. (Default)

ccsn_dc_estimate_mode

`< 0 | 1 | 2 >` Controls algorithm for estimating the end time for CCSN simulations. Default: `1`

- `0` Backward compatible with release `12.1`. The original algorithm for estimating simulation end times.
- `1` The end time is estimated from the previously simulated dc current results to get a more realistic end time. This helps in getting voltage waveforms that reach the final expected limits. (Default and recommended)
- `2` Use this setting to enable a simulation duration time estimation for ccsn output voltage simulations for use with some corner cases with large transition times.

Occasionally, the end time for voltage rise/fall CCSN simulations is not long enough and the final expected voltage was not reached. This parameter causes the estimated end time to be extended for such simulations.

This parameter must be set before the `char_library` command.

ccsn_dc_sweep_mode

< 0 | 1 | 2 | 3 | 4>

Controls whether CCSN characterization compatibility is available when running an external simulator.
Default (and Recommended): 4

0	External simulator compatibility setting for CCSN characterization.
1	(Default) Enables CCSN characterization compatibility when running an external simulator
2	Starts the CCSN DC sweep feature that helps in DC convergence. Use of this setting can speed up DC simulation and reduce the wall time tremendously. **Note:** When you set ccsn_dc_sweep_mode to 2, ensure that you are using SPECTRE 18.1 ISR6 or later.
3	Starts the CCSN DC sweep feature as is enabled when you set ccsn_dc_sweep_mode to 2. Setting ccsn_dc_sweep_mode to 3 however ensures that the output pin is placed in the outer loop, and the input pin is in the inner loop.
4	Starts the CCSN DC sweep feature as is enabled when you set ccsn_dc_sweep_mode to 2. Setting ccsn_dc_sweep_mode to 3 ensures that the input pin is placed in the outer loop and the output pin is in the inner loop, as it happens when ccsn_dc_sweep_mode=2. With the setting of 4, the sweep direction of the outer loop however depends on the input toggle state. For example, if the input wire is in falling state, the outer loop sweeps from 2*VDD to -VDD. In addition, the sweep direction of the inner loop depends on the output toggle state. For example, if the output wire is in falling state, the inner loop sweeps from 2*VDD to -VDD.

This parameter must be set before the char_library command.

ccsn_fanout_select_mode

`< 0 | 1 >` Sets CCSN groups on test pins. Default: `1`

When this parameter is set to `1`, Liberate includes CCSN groups on all test pins. Previously, all test pins did not had CCSN data.

This parameter must be set before the `char_library` command.

ccsn_io_skip_channel_inputs

`< 0 | 1 >` Changes the order in which the channel-connected wires are considered for probing when CCSN stages are determined. This has an effect on CCSN only when the *Inside View* algorithm cannot automatically drive CCSN characterization and the fallback approach of automatic vector determination is used. Default: `1`

- `0` No distinction about channel connectivity is made. This setting is used for backward compatibility with LIBERATE 13.1 ISR4 and prior releases.
- `1` Consider wires that are not channel-connected (does not connect to the transistor source or drain) first.

This parameter must be set before the `char_library` command.

ccsn_io_skip_channel_input_netlisting

`< 0 | 1 >` Enables pruning of channel connection input devices while simulating Channel Connected Blocks (CCBs).
Default: `1`

- `0` Resets the behavior to produce results that are backward compatible to Liberate 17.1 ISR1 and prior releases.
- `1` When the CCSN stages are being characterized without using *Inside View* algorithm (see `char_library -io`), ensures that these stages are simulated in the same way as they were pre-analyzed to derive stimulus by ignoring tracing through channel-related inputs in the CCB.

This parameter must be set before the `char_library` command.

ccsn_input_ccb_check

`< 0 | 1>` Controls whether the internally generated feedback should be recognized.
Default and recommended: `1`

- `0` Specifies not to recognize the internally generated feedback.

  Note: This setting is provided for backward compatibility with LIBERATE 15.1 ISR4 and earlier releases.
- `1` Enables recognition of internally generated feedback by using the logic to instantiate voltage sources.

This parameter must be set before the `char_library` command.

ccsn_input_xfr_probe_mode

`< 0 | 1 | 3>` Controls CCSN modeling of input pins connected directly to pass-gates.
Default: `1`
Recommended: `3`

- `0` Do not generate `ccsn_first_stage` groups for input pins that connect directly to transistor pass gates.
- `1` Generate `ccsn_first_stage_groups` for input pins that connect directly to transistor pass gates.
- `3` Enables extra analysis of pass gate channel control activity before writing out the spice deck. This is done to improve corner cases to sensitize the CCSN vector.

This parameter must be set before the `char_library` command.

ccsn_mega_mbit_hidden_mode

`< 0 | 1 | 2 | 3>` Controls how Liberate Mega mode (see mega_enable) generates hidden power arcs when characterizing CCSN for multi-bit FF cells. Default: `1`

- `0` This setting is provided for backward compatibility with LIBERATE14.1 ISR3 and prior releases.
- `1` Output hidden power arcs in sync with pin based ccsn_first_stage arcs in mega mode.
- `2` Uses faster method for determining CCSN vectors during preprocessing.
- `3` Do not attempt to restrict `input_ccb` vectors to hidden cases. Use this setting for speedup with large cells or with `input_ccb`/`output_ccb` models.

This parameter must be set before the `char_library` command.

ccsn_mega_mbit_skip_overlapping_vdata

`< 0 | 1>` Controls whether additional side-pin or vector settings should be skipped for cells where a partial WHEN state is sufficient to determine the scope of a CCSN stage.
Default: `1` (Skip overlapping vector data)

- `0` Do not skip CCSN simulations with overlapping vectors.
- `1` Skip overlapping vector data. Resulting libraries will be consistent with fewer simulations.

This parameter must be set before the `char_library` command.

ccsn_model_unbuffered_output

`< 0 | 1 | 2 | 3>` Specifies how to handle unbuffered outputs. Default: `3`

- `0` Do not model unbuffered output pins.
- `1` Cleanup or disconnect feedback loops on first stages of pins having output ports.
- `2` Cleanup or disconnect feedback loops on first stages of pins having output ports and enhanced handling of channel connected input/output CCBs for unbuffered pins.
- 3 In addition to the behavior in 2 above, effectively removes the influence of other CCBs when the port given as a side input is also affected by some other CCB's output. This setting ensures that a value is characterized for specific side input cases.

When a cell has an output driven through a transmission gate (it is unbuffered), and noise can propagate in the output and corrupt internal stored states, then it might be desirable to model this behavior in the CCSN noise constructs in the Liberty model.

This parameter must be set before the `char_library` command.

ccsn_pin_unconditional

`< 0 | 1 | 2 >` Controls whether Liberate should generate conditional pin-based CCSN stages.
Default: `0`

`0` Generates conditional pin-based CCSN stages.

`1` Stops Liberate from generating conditional pin-based CCSN stages.

When set to `1`, Liberate creates only a single CCSN pin stage when there are multiple CCSN pin stages possible. This can help to reduce the number of CCSN stages in an entire library by about 10%. Recent versions of PrimeTime require more complete CCSN models for improved correlation when noise-on-delay techniques are employed.

Note: The setting of `1` should be used only for backward-compatibility purposes.

`2` Instructs Liberate to use the worst pin-based CCSN stage as the default CCSN group. A worst CCSN pin-based group is selected from multiple pin-based CCSN candidates with `when` condition. This mode retains an LDB that includes conditional CCSN groups, but it affects the output library that only contains an unconditional default CCSN group.

Important

Set `ccsn_pin_unconditional` to `2` only when ccsn_use_io_ccb_format is set to `0`. This is required because the feature works only for a library that contains `ccsn_first_stage/ccsn_last_stage` groups, and does not works for a library that contains `input_ccb/output_ccb` groups.

This parameter must be set before the `char_library` command.

ccsn_prefer_two_sided_stages

`<0 | 1 | 2 | 3 | 4>` Controls whether Liberate should output two-sided (stage_type: both) CCSN stages whenever possible.
Default and recommended: `4`

This parameter helps to address the errors in 2008 versions of LC (LC200809-SP3).

- `0` This setting is available for backward compatibility. It restores the behavior prior to the 2.3 release.
- `1` Liberate always tries to generate CCSN stage that have *stage_type* set to *both* because `define_arc` sometimes splits the timing groups into separate rise and fall entities.
- `2` Liberate always tries to generate CCSN stages that have *stage_type* set to *both* and allow static check for rise and fall edges, regardless of whether the sensitization is feasible in the cell. Setting this parameter to `2` should address any LBDB-898 Synopsys LC error messages.
- `3` Allows handling of three-state cells when function-based CCSN `-io` mode is enabled.
- `4` Evaluates the vector sensitization inside CCB before merging the rise and fall vectors.

This parameter must be set before the `char_library` command is run.

ccsn_probe_mode

`< 0 | 1 | 2 >` Specifies the method for selecting a probe node for noise measurements. Default: `2` (Balanced)

- `0` Implements a strict interpretation of the specifications. Given a buffer where the wire between the two inverters is a complex RC network, this setting selects the input to one of the gates at the fanout of the RC network as the CCSN probe node.
- `1` Implements the sorting method.
- `2` Implements a balanced method that selects one of the nets on the RC network with a balanced RC path to both the pull-up (pmos) and pull-down (nmos) inputs as the probe node.

Controls how Liberate will select the observation or probe point, based on section 2.5 of the CCS Noise Characterization Guideline version 1.05, which details the considerations for selecting an observation point and handling parasitics during CCS Noise measurements.

This parameter must be set before the `char_library` command.

ccsn_redundant_pin_stages

`< 0 | 1 >` Set to duplicate certain CCSN stages on both arc and pin. Default: `0` (Do not duplicate)

- `0` Do not allow redundant stages. (Default and recommended)
- `1` Allow redundant stages (3.2p4_lcs behavior).

Prior to the 12.1 release, Liberate would duplicate some CCSN stages such that they would show up as both pin-based and arc-based CCSN groups. These may be considered as redundant groups, since the STA and SI tools have rules regarding priority of pin-based or arc-based noise stages.

This behavior was fixed in 12.1. For backwards-compatibility purposes, this parameter was introduced in 12.1.2.2 to allow the redundant stages.

This parameter must be set before the `char_library` command.

ccsn_skip_mod_vdata

`< 0 | 1 >` Liberate generates vecdata for CCSN arcs. Some of this vecdata is associated with tristate paths. The vecdata is global, and may be visible when building other CCSN stages. However, in some cases, the vecdata from tristate paths leads to false CCSN stages. Default: `1`

- `0` Get all matching vecdata when characterizing CCSN. The vecdata may originate from a tristate arc and then be modified for use with CCSN characterization. This setting is provided for backward compatibility with LIBERATE14.1 ISR3 and prior releases.
- `1` Ignore simulation vectors that originate from tristate arcs when characterizing CCSN.

This parameter must be set before the `char_library` command.

ccsn_unateness_based_merge_vecdata

`< 0 | 1 >` Adds unateness to the merging criteria for CCSN groups. Default and recommended: `1`

Note: Channel Connected Blocks (CCBs) for partially-defined WHEN states may result in vectors with mixed unateness. This can lead to undesirable results if merged into one CCSN group.

- `0` Do not consider unateness. This setting is provided only for backward compatibility with LIBERATE 16.1 ISR1 and prior releases.
- `1` Consider unateness.

This parameter must be set before the `char_library` command.

ccsn_unbuffered_output_mode

`< 0 | 1 >` Enables an algorithm that handles unbuffered outputs. Default: `1`

- `0` Disables the algorithm for backward compatibility. This setting can output a library with compilation issues that include a message about "Invalid DC Current".
- `1` Enables an unbuffered output stability check.

This parameter must be set before the `char_library` command.

cell_leakage_power_legacy_mode

`< 0 | 1 | 2 >` Determines when cell_leakage_power will be recalculated, based on the setting of `voltage_map`. Default: `0`

- `0` cell_leakage_power will be re-calculated for ***all*** voltage_map settings. (Default)
- `1` cell_leakage_power will ***not*** be re-calculated. (Behavior of release 3.1p1 and earlier.)
- `2` cell_leakage_power will be re-calculated ***only*** when `voltage_map=1`

Note: Intended for backward compatibility only.

This parameter must be set before the `write_library` command.

char_mos_term_cap_ski_mode

`< 0 | 1 >` Automates the MOS terminal capacitance calculation when using the SKI interface.
Default: `1`

- `0` Follow the setting of the `char_mos_term_cap` parameter. This setting is used for backward compatibility with LIBERATE 13.1 ISR4 and prior releases.
- `1` When the SKI interface is enabled (see `ski_enable`), and `char_mos_term_cap` is set to `-1`, then override the setting of `char_mos_term_cap` with a value of `2`. Otherwise, use the setting of `char_mos_term_cap`.

This parameter must be set before the `char_library` command.

conditional_expression_accuracy

`<0 | 1>` Merges state-dependent arcs into a single timing group (see conditional_expression). Use this parameter to improve the accuracy before merging the individual paths together.
Default: `1`

- `0` Uses coarse transistor resistance estimate so that different PVTs match. This setting is provided for backward compatibility with LIBERATE 15.1 ISR2 and prior releases.
- `1` Uses fine transistor resistance estimate for less merging and better `merge` and `separate` state matching (see conditional_expression).

This parameter must be set before the `char_library` command.

constraint_dependent_nonseqsetuphold

`< 0 | 1 | 2 >` Controls characterization of the `non_seq_setup` and `non_seq_hold` categories of data.

- `0` Do not characterize dependent `non_seq_setup` and `non_seq_hold`.
- `1` Recharacterize `non_seq_setup` based on `non_seq_hold`.
- `2` Recharacterize `non_seq_hold` based on `non_seq_setup`.

This parameter must be set before the `char_library` command.

constraint_search_time_reltol_mode

`< 0 | 1 | 2 >` Affects when bisection will stop. For backward compatibility only. Default: `2`

- `0` For backward compatibility with pre-3.2 versions. Use of this setting might result in more search iterations. (Not recommended.)
- `1` For backward compatibility with pre-13.1.3 version. Use of this setting might result in more search iterations. (Not recommended.)
- `2` The search tolerance follows `constraint_search_time_abstol` and `constraint_search_time_reltol`. (Default)

For pre-3.2 versions of Liberate, if the constraint is less than 1ps, then bisection will stop only if the search window is `< 1ps` or `constraint_search_time_abstol`, whichever is smaller.

For version prior to LIBERATE 13.1.3, this occurs only when `constraint_bisection_mode=2`. For versions post LIBERATE 13.1.3, this does not occur.

This parameter must be set before the `char_library` command.

constraint_slew_degrade_nominal_check

`< 0 | 1>` Controls whether the nominal slew should be adjusted if `constraint_slew_degrade` and `constraint_delay_degrade_nominal_check` are enabled (`>0`), and the nominal delay is adjusted.
Default: `1`

- `0` Provides backward compatibility with the LIBERATE 18.1 ISR1 and prior releases.
- `1` Adjusts the nominal slew if the following conditions are true:
  - constraint_slew_degrade is enabled (`>0`)
  - `constraint_delay_degrade_nominal_check` is enabled (`>0`)
  - the nominal delay is adjusted

  For more details about the check algorithm, see constraint_delay_degrade_nominal_check.

This parameter must be set before the `char_library` command.

constraint_stim_precision_auto_adjust

`< 0 | 1>` Allows Liberate to write out more than 8 digits of precision in SPICE decks.
Default: `1`

- `0` Maintains a fixed number of digits. This setting is provided for backward compatibility with the LIBERATE 16.1 and prior releases.
- `1` Auto-expands the digits to maintain the needed precision.

Note: For most simulations, the difference between the largest and smallest events might be contained within 8 digits of precision. For extremely long simulations, more than 8 digits might be necessary. The bound is generally applicable with transient simulations times of `1e-4` or larger.

This parameter must be set before the `char_library` command.

default_non_unate_rcvr_cap_adjust

`< 0 | 1 >` Sets the receiver cap groups to follow the input pin direction. Default: `1`

- `0` The receiver cap groups will follow the output pin direction. (Backward compatibility setting.)
- `1` The receiver cap groups will follow the input pin direction for non-unate arcs. (Default)

This parameter must be set before the `char_library` command.

default_power_subtract_hidden_mode

`< 0 | 1 | 2 >` Controls `subtract_hidden_power` algorithm. Default: `2`

- `0` Use the algorithm from `3.2`. The default `internal_power` is chosen and then the `hidden_power` is subtracted.
- `1` Use the algorithm from `12.1 ISR1`. The `hidden_power` is subtracted and then the default `internal_power` is selected.
- `2` Use the algorithm as of `12.1 ISR4`. This is a fix to mode 1 where hidden power is sometimes not subtracted. (Default)

This parameter is used to control the algorithm used when `subtract_hidden_power` is enabled.

This parameter must be set before the `write_library` command.

default_timing_tristate_enable

`< 0 | 1 >` Selects the default group timing type for `three_state_enable` arcs. Default: `1` (default arc `timing_type` uses `three_state_enable`)

- `0` The default arc timing_type for three_state_enable arcs will be set to 'combinational'. This setting is provided for backward compatibility with release LIBERATE 13.1 ISR4 and prior releases.
- `1` The arc timing attribute uses `three_state_enable` (Default).

This parameter is used to control `timing_type` attribute for default `three_state_enable` timing groups.

This parameter must be set before the `char_library` command.

def_arc_delay_metric_mode

`< 0 | 1 >` For backward compatibility with `define_arc -metric`. Default: `1`

- `0` Restores the behavior of the LIBERATE 12.1 and prior releases where under certain conditions, the Liberate internal algorithms would override the user-specified `define_arc -metric`.
- `1` (Default and recommended.)

Note: The `define_arc -metric` option should always control the constraint measurement metric.

This parameter must be set prior to the `char_library` command.

disable_current_measure_effort

`< 0 | 1 >`	Instructs the tool to interpolate the actual crossing time of output current crossing the current degradation threshold. Default: `1`

The `three_state_disable` delay measurement interpolates between the time steps reported by the circuit simulation. Set this parameter to `0` to restore the behavior of the 2.5 and prior releases where Liberate reports the time point when output current drops below the current degradation threshold.

This parameter must be set before the `char_library` command.

driver_waveform_lib_mode

`< 0 | 1 >`	Affects the way driver waveforms are stored in the library in the `normalized_driver_waveform`. Pertains to the falling waveform portion only. Use this parameter for backward compatibility. Default: `1`
`0`	Format for normalized_driver_waveform used with version `3.2p3` or earlier.
`1`	Standard format for normalized_driver_waveform. (Default and recommended.)

This parameter must be set before the `write_library` command.

ecsmn_allow_duplicate_condition

`< 0 | 1 >`	Controls modeling of duplicate ECSMN groups. Default: `0`
`0`	Filters duplicate ECSMN groups. If duplicate conditions exist, the worst case is selected by checking the data.
`1`	Allows duplicate ECSMN groups for backward compatibility.

This parameter must be used before the `char_library` command.

ecsmp_invert_gnd_current

`< 0 | 1 >` Inverts current values for ecsm_current_waveform groups.
Default: `1` (Invert current)

- `0` Do not invert currents.
- `1` Inverts the current values for compatibility with EPS. (Default)

When reporting current values, Liberate usually follows the convention where positive currents flow into the cell and negative current flow out of the cell. In this fashion, supplies are normally positive and grounds are normally negative.

Cadence power tools prefer to have ground currents reported as positive values. This parameter controls whether or not to invert the current on grounds, which are reported in index_1 of ecsm_current_waveform groups.

This parameter must be used before the `char_library` command is run.

em_check_lib_data

`<0 | 1>` Controls the printing of messages that notify you if any electromigration (EM) data for a pin or an arc is missing when the merge_library command is run.
Default: `1`

- `0` Does not print the messages.

  Note: This setting is used for backward compatibility with the releases prior to LIBERATE 19.2.
- `1` Prints the messages.

This parameter must be used before the `merge_library` command is run.

Example

```
set_var -cell CELLNAME2 em_check_lib_data 1
```

The parameter setting given above will ensure that when the `merge_library` command is run, Liberate will print messages such as following if any EM data is missing in cell, `CELLNAME2`, in the original `.lib` file:

```
Info (merge_library) : Merging library em.lib into nom.lib
Info (merge_library) : Merging .lib for cell CELLNAME from em.lib
Info (merge_library) : Merging .lib for cell CELLNAME2 from em.lib
WARNING (LIB-364): (merge_library): No electromigration (EM) data found in cell:
'CELLNAME2' for the arc: related pin: 'R', pin: Q_F' in merged library. If this
data is required, provide the missing EM data.
```

em_check_table_data

`<0 | 1>` Controls whether the electromigration (EM) tables should be checked for valid non-zero entries.
Default: `1`

- `0` Use this setting for backward compatibility with the LIBERATE 18.1 ISR2 and prior releases where EM tables are not checked.
- `1` Checks the EM tables for valid non-zero entries and flags the errors when appropriate.

This parameter must be used before the `char_library -em` command is run.

em_unmatched_resistors_check

`< 0 | 1 >` Enables the checks for unmatched resistors during electromigration (EM) characterizations.
Default and recommended: `1`

- `0` Disables the checks. This setting is provided for backward compatibility.
- `1` During EM simulations, Liberate generates warning messages when Voltus-Fi fails to find the rules for resistors.

This parameter must be used before the `char_library` command.

extsim_model_include_multi_vector_mode

`< 0 | 1 >`	For backward compatibility. Uses 3.1 (and prior) algorithm Default: `1`
	`0` Backward compatible with release `3.1`.
	`1` (Default)

Version 3.1p1 corrects an issue handling constraint-related vectors in the EMI flow when `extsim_flatten_netlist` is set to 0.

extsim_save_driver

`< 0 | 1 >`	Saves SPICE decks for failing active driver simulations. Default: `1`
	`0` Specifies not to save any active driver simulation decks. This setting is available for backward compatibility.
	`1` Save 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` parameters. If the saving of decks is not enabled using these parameters, no decks are saved.

This parameter is used to enable the saving of simulation decks used by the `set_driver_cell1` command to characterize the active driver output waveform simulation decks.

This parameter must be set before the `char_library` command.

floating_node_consistency_check

`< 0 | 1 >` This parameter enables floating node consistency checks introduced in 3.1p2. Default: `1`

- `0` Backward compatible with release `3.1p2`. Disables floating node consistency checks.
- `1` Enable the floating node consistency checks. (Default and recommended.)

This parameter should only be disabled to reproduce the inconsistent floating node handling behavior of Liberate versions 3.1p2 and earlier for regression purposes.

Note: This behavior has known issues and is not recommended for general use.

This parameter must be set before the `char_library` command.

ignore_dummy_fets

<0 | 1 | 2 | 3 | 4> Specifies method for dealing with PODE devices. Default: 2

0	The *Inside View* algorithm considers the dummy devices to have an impact on the functionality of the circuit. Previously, the gates to these devices were viewed as CCC inputs. When the gates were floating, they would generate two vectors, one for each value of 0/1 for each floating gate. When many of these devices are present in a cell, it results in an exponential increase in the number of vectors to evaluate or simulate for no real purpose. This can significantly increase runtime as the values of the nodes associated with these devices is considered during pre-processing and vector generation and differ only in how initial conditions (.ic) are set for the dummy devices with floating gates. This setting is provided for backward compatibility with LIBERATE 13.1 and prior releases.
1	Ignore functional impact. Set each floating gate to its disabled state. Multiple floating gates on the same simwire may be assigned different initial conditions (.ic).
2	Ignore functional impact. Set each floating gate to its disabled state. If multiple parasitic devices with floating gates are on the same simwire, a single .ic is assigned to the wire. (Default and Recommended)
3	Ignore functional impact. Do not assign any .ic to floating gates. SPICE simulators that are not capable of handling dummy devices natively may have problems with DC-convergence if this setting is used and is therefore not recommended.
4	Transistors with disconnected gates are treated as off for function analysis and left floating in the simulation.

Use this parameter to determine how the *Inside View* algorithm handles non-functional or

"dummy" parasitic transistors such as PODE devices. These are devices added for parasitic reasons and that have "pode" in the model name of the device.

This parameter must be set before the `char_library` command.

mega_compatibility_mode

`< 0 | 1 | 2 | 3 | 4 | 5 >`

Enables the algorithm to make the vector selection process backward compatible with the implementation available in releases prior to LIBERATE 18.1.
Default: `5`

From time to time, enhancements are made in vector selection for multi-bit and large combinational cells to avoid problems due to the exponential number of possible vectors. These changes can alter the subset of vectors returned (especially for hidden arcs). Therefore, this parameter has been provided for backward compatibility with the required old implementations.

`0`	Provides backward compatibility with the implementation available in LIBERATE 15.1 ISR5.
`1`	Provides backward compatibility with the implementation available in LIBERATE 16.1 base release.
`2`	Provides backward compatibility with the implementation available in LIBERATE 16.1 ISR1.
`3`	Provides backward compatibility with the implementation available in LIBERATE 16.1 ISR2.
4	Provides backward compatibility with the implementation available in LIBERATE 17.1 ISR3.
5	Provides backward compatibility with the implementation available in releases prior to LIBERATE 18.1. This is the default setting.

This parameter must be used before the `char_library` command.

model_operating_condition_mode

`< 0 | 1>` Controls whether the `set_operating_condition` command will control the `nom_voltage` in the `operating_condition` group that exists in the output library.
Default: `1`

- `0` The `nom_voltage` in the output library's `operating_condition` group follows the `set_vdd` setting. This setting is provided for backward compatibility.
- `1` The `nom_voltage` in the output library's `operating_condition` group follows the value of the `set_operating_condition -voltage` command.

This parameter must be used before the `char_library` command.

nonseq_as_recrem_char_mode

`<0 | 1 | 2>` Specifies how the `non_seq_setup` and `non_seq_hold` categories of data are handled in Liberate when they are specified with the `char_library -skip` command. Default: `2`

- `0` Maps the `non_seq_setup` and `non_seq_hold` categories to the `recovery` and `removal` categories.

  Note: Use this setting for backward compatibility with the LIBERATE19.1 and prior releases.
- `1` Handles the `non_seq_setup` and `non_seq_hold` categories separately from the `recovery` and `removal` attributes.
- `2` Handles the `non_seq_setup` and `non_seq_hold` categories separately from the `recovery` and `removal` attributes. In addition, the `set_var -type` command supports `non_seq_setup` and `non_seq_hold`. The saved decks and messages related to `non_seq_setup` and `non_seq_hold` will reference `non_seq_setup` and `non_seq_hold` instead of `recovery` and `removal`.

This parameter must be set before the `char_library` command is run.

non_seq_probe_mode

`< 0 | 1 >` Controls the probe selection mode for *nonseq_setup/hold.* Default: `1`

- `0` Compatibility with releases prior to version 2.2.
- `1` Compatibility with release 2.2 and later. (Default)

Specifies the non_seq probing order. When set to `1`, Liberate will try to honor `constraint_output_pin`.

This parameter must be set before the `char_library` command.

parse_sfe_parser_mode

`< 0 | 1 | 2>` Selects the parser that should be used to read in the netlist and models files specified using the read_spice command.
Default: `2`

- `0` Switches back to the old behavior (that is, LIBERATE 12.1 parser) where the native parser is used for all netlist formats.
- `1` Uses Spectre Front End (SFE) to parse Spectre format netlist and model files when the `read_spice -format spectre` command is used.

  Note: Set `parse_sfe_parser_mode` to `1` if you want to switch to the behavior that was available in LIBERATE 12.1 to LIBERATE 16.1 ISR4 releases.
- `2` Uses SFE for parsing SPICE and Spectre format netlists and models.

This parameter must be set before the `read_spice` command is run.

power_multi_vector_mode

`< 0 | 1 | 2 | 3 >` Enables different algorithms for vector selection related to an issue where Liberate can choose a wrong vector when an arc contains multiple vectors and the voltage_map parameter is greater than `0`. This can produce power results that might not be worst case. Default: `2`

- `0` Backward compatible with release 3.1p1.
- `1` Corrects the issue of choosing a wrong vector. Backward compatible with release LIBERATE 13.1 ISR1.
- `2` Always update composite current source power (CCSP) energies when power results are updated.
- `3` Enables CCSP selection to follow Non-Linear-Power-Model (NLPM) selection.

This parameter must be set before the `char_library` command is run.

power_subtract_leakage_tran_mode

`< 0 | 1 >` Specifies whether Liberate should check individual simulation duration for multiple internal vectors, or use one reference. Default: `1` (Check individual vectors)

- `0` Use the same simulation duration for all internal vectors. (For backward compatibility only.)
- `1` Use individual simulation duration values for each vector. (Default and recommended.)

This is a backward compatibility parameter. Prior to version 3.2p3, Liberate uses the same simulation duration for leakage subtraction for all internal vectors. This can produce inconsistent power results. Set this to 1 to use individual simulation duration values for each vector.

This parameter must be set before the `char_library` command is run.

reset_negative_power_mode

`< 0 | 1 >`	Controls the behavior for fixing negative power values. Default: `1` (Follow `reset_negative_power`)
`0`	Do not reset negative power values for bulk rails. (For backward compatibility only)
`1`	Follow `reset_negative_power` for all supplies. (Default and recommended)

When set, Liberate follows `reset_negative_power` for bulk nodes. The recommended setting is `1`. This is because it allows `reset_negative_power` to be applied to all supplies.

This parameter must be set before the `char_library` command is run.

set_pin_slew_threshold_mode

`< 0 | 1 >`	Controls handling of slew threshold derating for power. Default: `1`
`0`	Scale the `index_*` and value in the pin group specified. (Only for backward compatibility with LIBERATE 13.1 ISR2)
`1`	If the pin specified in the `set_pin_threshold` command is the arc `-pin`, then scale only the rise/fall_transition value. When it is the `-related_pin`, then scale only the slew index, which is usually `index_1`. (Default)

Note: This parameter enables correct processing of the `set_pin_threshold` command settings.

This parameter must be set before the `char_library` command is run.

si_write_output_voltage_compatibility_mode

`<0 | 1>`	Creates an `output_voltage` group in the `merge_library` flow when the input library has a template that includes `variable_1: iv_output_voltage`. Default: `1`

`0`	For backward compatibility. Creates an `output_voltage` group in the final merged library.
`1`	Creates an `output_voltage` group in the final merged library. If the input library does not have the `output_voltage` group, the `output_voltage` group will not be generated for the final merged library and an error message is printed.

This parameter must be set before the `merge_library` command.

simultaneous_switch_from_cell_when

`< 0 | 1 | 2 | 3>` Enables automatic dual input switching.
Default: `2`

`0`	Do not look for simultaneous switching side pins. **Note:** This setting is for backward compatibility to the LIBERATE 14.1 ISR5 release.
`1`	The simulation vector must meet the `define_cell -when` and `define_arc -when` states both before and after the `related_pin` switches. This usually requires side pins to switch simultaneously with the related pin. The simultaneous switching vector algorithm is not applied to constraints or MPW with `define_arc`. **Note:** This setting is for backward compatibility to the LIBERATE 18.1 release.
`2`	Enables selection of additional switching input pins for a `define_arc` command, if necessary to satisfy the `define_cell -when` condition.
`3`	Ensures that the `define_cell -when` command is correctly accounted for. In addition, the `define_arc -when` command includes the `define_arc -type power` command for all cases.

This parameter must be used before the char_library command.

switch_cell_dc_partition_include_primary_input

`< 0 | 1 >` Controls the partition used to characterize the `dc_current` for power switch cells when the CCB based format is enabled (see ccsn_use_io_ccb_format).
Default: `0`

- `0` Limits the partition to only include CCB devices and consequently, end at internal nodes.
- `1` Includes non-CCB related devices so that both the primary control pin and the switched rail supply pin can be in the partition.

This parameter must be set before the `char_library` command is run.

switch_cell_infer_unateness_from_ldb

`< 0 | 1 >` Controls determination of unateness for switch cells.
Default and recommended: `1`

- `0` Enables behavior where unateness of switch cells was always negative. *Not recommended.*
- `1` Unateness of switch cells is correctly determined and written to the library.

This parameter is provided for backward compatibility when writing out a library from an older LDB (prior to v3.2). It is relevant only for switch cells.

This parameter must be set before the `write_library` command is run.

tristate_pin_cap_use_arc

`< 0 | 1 | 2 >` Default: `2`

`0`	For backward compatibility.
`1`	New method only.
`2`	New method with fall back to original method if required. (Default)

A change has been made in the way input capacitance is computed for tristate outputs and bidi pins. Previously, separate simulations were created to compute rise and fall capacitance. The state of side inputs for the simulations was determined from the `three_state` attribute and otherwise set arbitrarily. This could in some cases result in a measurement while the pin was enabled as an output which would give a very large value.

The new behavior is to not generate the extra simulations if input capacitance is already available from hidden or delay measurements. If such measurements are not available, as for tristate outputs, then any tristate enable measurements are used to determine a state to measure in which the output is hi-Z. Only if no tristate enable measurement is specified is the original construction used.

user_data_keep_simple_attr_quotes

`< 0 | 1 >` Specifies whether the quoted string value that defines the attribute name in the user data should be wrapped within quotes. Default: `1`

`0`	Old behavior for backward compatibility. Liberate reads in the value of an attribute from the user_data file and removes the quotes in the internal database.
`1`	If the value of an attribute in the user_data file has quotes, the attribute in the new library will also have quotes.

If `user_data_quote_attr` includes the attribute name and/or the `user_data_quote_simple_attr` is set to `1`, the attribute value is enclosed within quotes. This happens even if the value of the attribute has no quotes in the user_data file.

This parameter must be set before the `write_library` command is run.

vec_data_sort_mode

`< 0 | 1 >` Controls whether CCSN models should be changed to reduce the dependence of CCSN models on arbitrary internal vector ordering. Default: `1`

Note: This parameter provides backward compatibility with the releases prior to the LIBERATE 16.1 base release.

- `0` Allows CCSN models to change due to unrelated and irrelevant changes in other parts of the code.
- `1` Restricts changing of CCSN models due to internal ordering of vectors.

This parameter must be set before the `char_library` command is run.

voltage_map_ldb_char_mode

`< 0 | 1 >` Corrects erroneous supply name in library. Default: `1`

- `0` Backward compatible with version `3.2p3` and earlier.
- `1` Corrects issue where default supply name was erroneously written to library when `voltage_map=2`. (Default)

Applicable to circuits that do ***not*** use VDD, while running `read_library` / `write_library` flow.

This parameter must be set before the `write_library` command.

write_logic_function_async_mode

`< 0 | 1 >` Controls whether an arc with a *timing_type* of *preset* and *clear* should be changed to a *timing_type* of *combinational* based on the value to which the `write_logic_function` parameter is set. Default: `1`

- `0` When the `write_logic_function` parameter is set to `0`, the arcs with a *timing_type* of *preset* and *clear* will be changed to a *timing_type* of *combinational*. This setting provides backward compatibility with LIBERATE13.1 and prior releases.
- `1` The *timing_type* of *preset* and *clear* will not be affected by the value set for the `write_logic_function` parameter.

This parameter must be set before the `write_library` command.

Legacy Debug Parameters

The parameters discussed in this section may or may not work. These are a list of parameters that were used for development or debug purposes. **Use at your own risk!**

ccs_retry_info

`< 0 | 1 >`	Enable printing of CCS retry criteria details. Default: 0 (don't print)

Setting this parameter to a 1 will turn on printing of CCS retry criteria information into the log file. The information printed will include the cell name, pin, related pin, WHEN condition, transition and criteria. This option has no effect if `ccs_retry_mode=0`.

This parameter must be set before the `char_library` command.

Example

```
*Warning* (char_library) Arc cell=IIND4D2LVT pin=ZN related_pin=A2 when=
fall_transition have the following issues:
    Failed ccs_voltage_tail_tol criteria
```

ccs_retry_mode

`< 0 | 1 | 2 | 3 >`	Enables CCS simulation retry methodologies. Default: `0` (Disabled)

`0`	Disabled.
`1`	Check `ccs_retry_multi_peak_tol` criteria and `set_ccs_retry_thresholds` against `ccs_abs_tol` and `ccs_rel_tol` values.
`2`	Check criteria from option `1` and `ccs_retry_voltage_tail_tol`.
`3`	Check `ccs_retry_voltage_tail_tol` criteria only.

In certain cases the SPICE simulation options specified through `alspice_option`, `extsim_option`, or `extsim_ccs_option` may not have sufficient accuracy or can result in trapezoidal oscillation in the current waveform. Certain versions of third-party downstream

tools may poorly handle an `output_current` waveform with this oscillation, resulting in correlation errors.

Liberate can automatically detect these oscillations and re-simulate with higher accuracy SPICE settings. This parameter controls the methodology and criteria used for detecting failures.

Note: Enabling this methodology will have an impact on runtime and will likely result in many re-simulations. If used, only option 3 is advised. Prior to enabling this functionality, it is recommended to use the following options and check for CCS correlation:

- `ccs_init_voltage_comp_thresh 1.1`
- `ccs_voltage_tail_tol 0.951`
- `ccs_voltage_tail_trim_tol 0.999`

Depending on the version or settings of your versions of Library Compiler and STA tools, setting `ccs_voltage_tail_tol` to `0.981` might achieve better results.

See also:

- `ccs_retry_info`
- `ccs_retry_multi_peak_tol`
- `extsim_ccs_retry_option`
- `extsim_ccs_retry_tran_append`
- `set_ccs_retry_thresholds`

The default and recommended setting is 0.

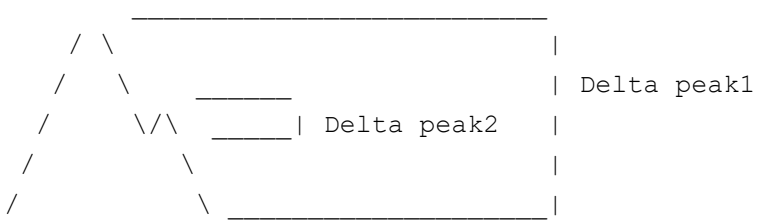
This parameter must be set before the `char_library` command.

ccs_retry_multi_peak_tol

<value>	Threshold for CCS waveform multi-peak failure criteria. Default: 0.05 (5% of peak)

If ccs_retry_mode is enabled, then Liberate will evaluate the CCS current waveform for multiple peaks. If multiple peaks are detected, then Liberate find the height of each peak. If the ratio between peaks is less than this parameter, then this peak is not flagged for failing criteria.

ccs_retry_multi_peak_tol : 2% tolerance for multi-peak. If (delta peak2) / (delta peak1) < 2%, this is not a failure:



The default and recommended setting is 0.05

This parameter must be set before the `char_library` command.

ccs_retry_voltage_tail_tol

<value>	The stopping point as a ratio of supply of the CCS waveform for CCS retry criteria. Default: `0.981` (follow `ccs_voltage_tail_tol`)

For CCS timing waveform data, if ccs_retry_mode 2 or 3 are enabled then the tail of the integrated v(t) obtained from the sampled i(t) is checked against ${ccs_retry_voltage_tail_tol} * supply_swing. Failing checks result in simulation retries with higher accuracy SPICE options. Default: 0.981 (follow ccs_voltage_tail_tol; output must swing to within 0.019% of supply).

This parameter must be set before the `char_library` command.

Example:

```
set_var ccs_retry_voltage_tail_tol 0.951
```

extsim_ccs_retry_option

<"options">	Options to be used for CCS timing recharacterization with external SPICE. Default: set to `extsim_ccs_option`

This parameter specifies the list of options to be used by the external SPICE simulator when characterizing CCS timing. This command only applies when the ccs argument is used with `char_library` and when `ccs_retry_mode>0`.

This option set will supersede any settings in `extsim_ccs_option`. If used, this parameter should contain options that will yield higher accuracy and control trapezoidal oscillation

present in a current waveform in a SPICE simulation. The option string is passed as an .option line in the SPICE decks Liberate creates for characterization.

See also `ccs_retry_mode` and `extsim_ccs_retry_tran_append`.

This parameter must be set before the `char_library` command.

Example:

```
# Set SPICE options for CCS retry simulations
set_var extsim_ccs_retry_option "runlvl=6 accurate method=BDF"
```

extsim_ccs_retry_tran_append

`<"options">` Additional options to append to .tran. Default: "" (none)

This parameter is used to add extra options to the .tran statement during `ccs_retry` simulations. If Spectre is used, it is recommended to set this option.

See also `ccs_retry_mode` and `extsim_ccs_retry_option`.

This parameter must be set before the `char_library` command.

Example:

```
# Set conservative mode for Spectre for CCS retry simulations
set_var extsim_ccs_retry_tran_append "errpreset=conservative"
```

G

Glossary

Glossary of common industry terms, and other specialized terms used in this manual.

AOCV	Advanced On-Chip Variation
BIST	Built-in Self Test
bsub	Batch Submission (part of LSF)
CCB	Channel Connected Blocks
CCC	Channel Connected Component (region)
CCR	Channel Connected Regions
CCS	Composite Current Source (Synopsys format)
CCSN	Composite Current Source, Noise
CCSP	Composite Current Source, Power
CCST	Composite Current Source, Timing
CSM	Current Source Models
DCOP	Desktop Communication Protocol
DFM	Design for Manufacturing
DSPF	Detailed Standard Parasitic Format
ECSM	Effective Current Source Model (Cadence format)
EIA	Electronics Industries Alliance (Standards body)
EMI	External_sim Model Include / Electro-magnetic Interference
JMS	Job Management System
LSF	Load Sharing Facility (job scheduler)

MLD	Measurement Description Language (Spectre)
NLDM	Non-Linear Delay Model
OCV	On-Chip Variation
OMC	Open Modeling Coalition
PCR	Program Change Request
PVT	Process, Voltage, Temperature (Corner analysis)
PWL	Piece-Wise Linear
qsub	Queue Submission
SDF	Standard Delay Format
SGE	Sun Grid Engine (job scheduler)
SI	Signal Integrity
SKI	Spectre Kernel Interface
SPEF	Standard Parasitic Extraction Format
SSTA	Statistical Static Timing Analysis
STA	Static Timing Analysis
TCAM	Ternary Content-Addressable Memory
VCS	Verilog Compiled-code Simulator
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VITAL	VHDL Initiative Towards ASIC Libraries