cadence®

Virtuoso Liberate Reference Manual

Product Version 14.1
November 2014

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

Contents

5
Liberate Parameters

1

Introduction

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

System Requirements

The programs Liberate, Liberate MX, Liberate LV, Variety, and ALAPI run exclusively on Linux Operating Systems. Below is a listing of supported platforms:

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

For detailed information about the requirements, see Computing Platforms.

The Role and Importance of Libraries

Creation of electrical views is a necessary pre-requisite 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 final silicon.

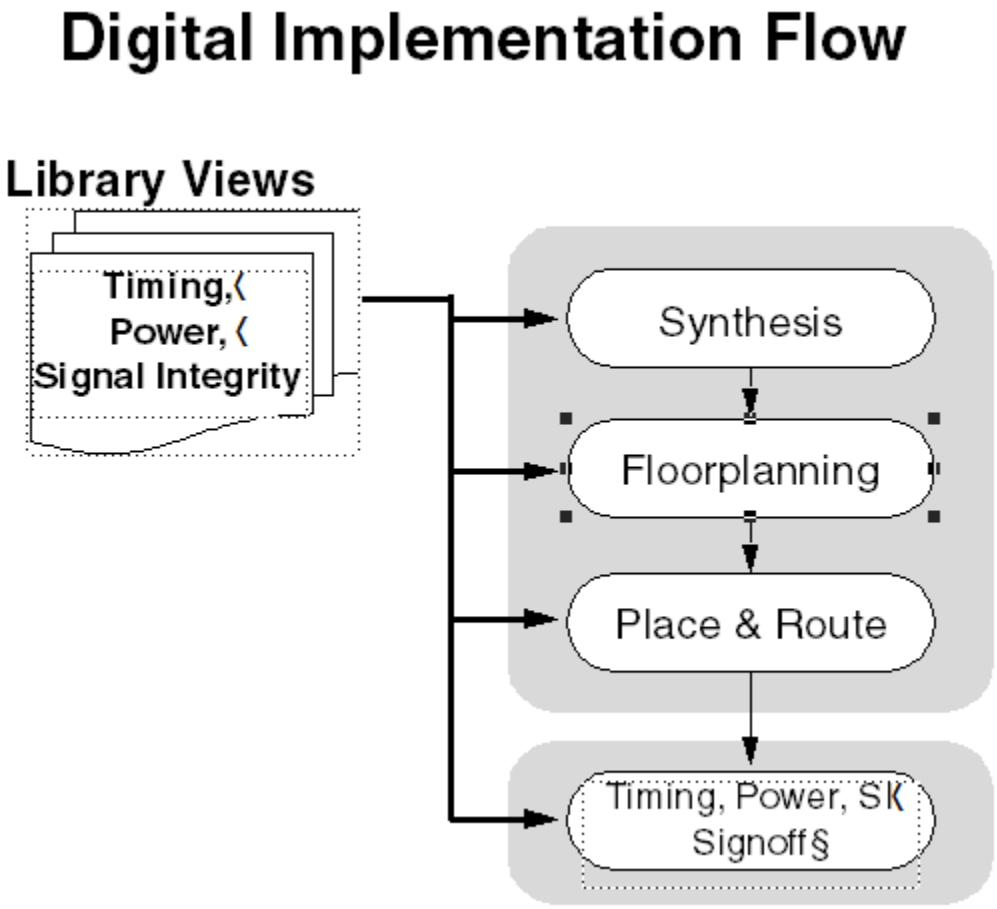


Figure 1: The Digital Design Implementation Flow

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.

A Growing Problem

In nanometer geometries (65nm or below) the required number of library views is growing dramatically because of issues related to leakage power and process variation. To minimize leakage power at deep submicron nodes, we see process variations such as LVT, RVT, and HVT (low/regular/high Vt) 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 for 130nm.

Nanometer Requirements	**130 nm**	**65 nm**	**28 nm**
PVT	3 (fast,typ,slow)	3 (fast,typ,slow)	3 (fast,typ,slow)
Leakage	n/a	3 Vths	3 Vths
Voltage Scaling	n/a	2 Voltages	2 Voltages
Temperature	n/a	n/a	2 Temperatures
Yield	n/a	n/a	2 Yield
Total Views	**3**	**18**	**72**

Figure 2: Library Views Required for Nanometer Process Nodes

In addition, at 28nm, it will be common to use different physical libraries that tradeoff performance for yield, where the higher-yielding libraries will use feature sizes that are greater than minimum. At 28nm, on-chip temperature gradients and temperature inversion effects will also need to be accounted for. This combination will lead to a further increase in library views, potentially 24 times or more than those for 130nm.

Performance is Key

Given this 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 runtimes. 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 **Alspice**, Liberate supports using external SPICE simulators such as **HSPICE®, Eldo®** and **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 Liberate's speed advantage.

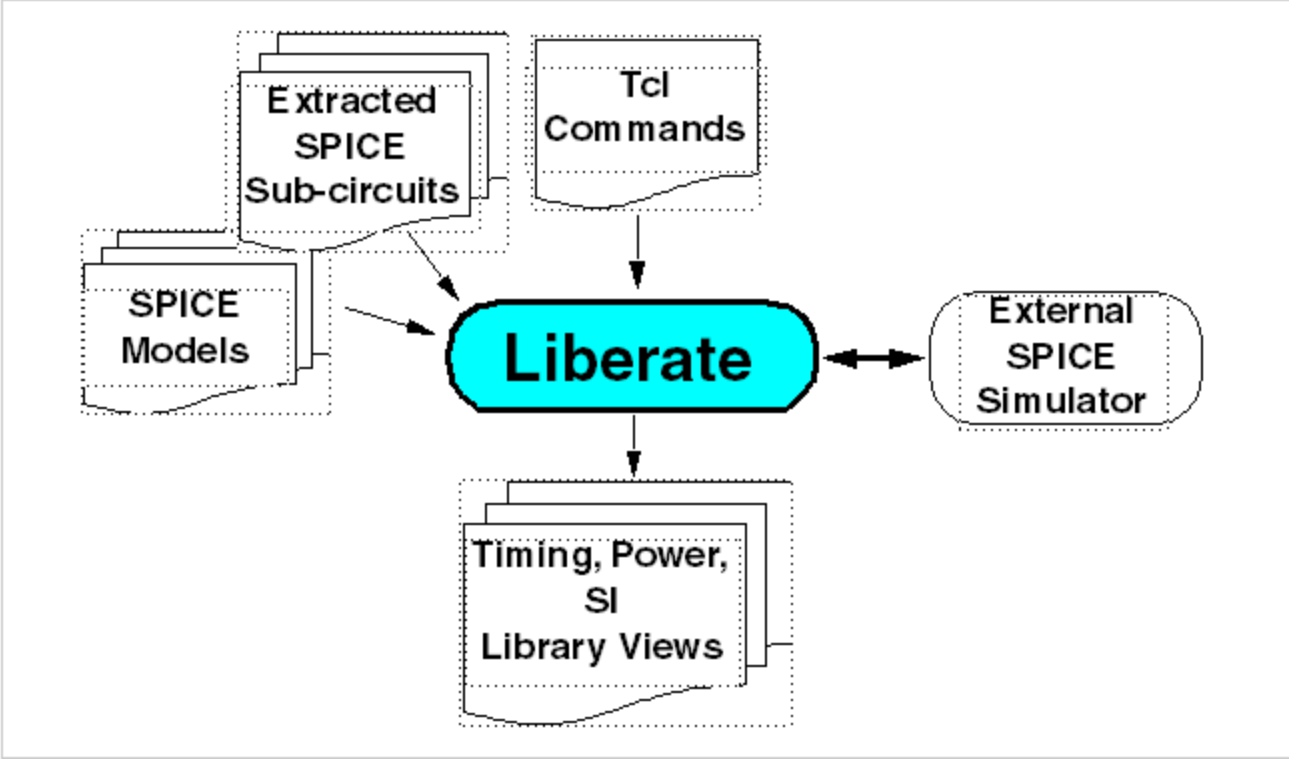


Figure 3: Using Liberate with an external SPICE simulator

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, make sure that it is installed correctly and that all the necessary pre-requisite data is available. (See the **Cadence Installation Guide**, and **Cadence License Manager** manuals.)

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)
```

64-bit Machine Support

Liberate also ships with support for 64-bit machines. To use a 64-bit port, set the **ALTOS_64** environment variable before running Liberate, as shown below.

```
% setenv ALTOS_64 1
```

If you are using Spectre Kernel Interface (SKI), the Liberate and Spectre binaries must be compatible. This is defined as the same style of binary (32- vs 64 bit) and version. For the version of MMSIM (also known as, Spectre) that is tested for compatibility with Liberate, see the README which ships with the Liberate release. Do not use a version of MMSIM older than the tested compatible version. To enable 64-bit support in MMSIM:

```
setenv CDS_AUTO_64BIT 1
```

Shell Environment Variables for Controlling Licensing Checks

- ALTOS_LIC_MAX_TIMEOUT
  `setenv ALTOS_LIC_MAX_TIMEOUT <value>`

where,

`value` is time in seconds

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

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

- ALTOS_LIC_CHECK_ALT_TIMEOUT

```
setenv ALTOS_LIC_CHECK_ALT_TIMEOUT <value>
```

where,

`value` is time in seconds

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

Server-Client Licensing

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

- 1 characterization run using 40 client processes
- 2 simultaneous characterization runs, each with 20 client processes
- 2 simultaneous characterization runs, 1 with 30 client processes, 1 with 10 client processes

Wait for Available License

When a Liberate job is submitted, a request is made for a license. To request that Liberate wait until a license becomes available, it is necessary to set the following CSH environment variable:

```
setenv ALTOS_QUEUE 1
```

Liberate clients run using different types of client license features, depending on the product names. Some product licenses can be mixed and matched together. Liberate clients can run using the following product licenses: Liberate_Client, Liberate_LX_Client, and Variety_LX_Client. For example, to run Liberate with 4 threads, two Liberate_Client and 2 Liberate_LX_Client features could be utilized.

When a Liberate server starts, it checks out 1 server license – Liberate_Server, Variety_LX_Server, or Liberate_LX_Server. Later, when simulations are ready to begin, Liberate tries to check out N clients, where N is the number of threads specified in the char_library -thread command option. Liberate tries to check out a combination of:

Liberate_Client + Liberate_LX_Client + Variety_LX_Client

- If Liberate acquires all N licenses, then it starts simulations using N threads.
- If Liberate acquires M licenses, 0 < M < N, and ALTOS_QUEUE is not set to 1, then it starts M thread of jobs.
- If Liberate does not acquire a license and ALTOS_QUEUE is not set to 1, then it quits.
- If Liberate acquires fewer than N licenses and ALTOS_QUEUE is set to 1, then it waits for up to the value of the lic_max_timeout variable, trying to get all N licenses. After lic_max_timeout is reached, if Liberate acquires M licenses and M > 0, then it starts M threads.
- If M is 0, then it again waits for another lic_max_timeout seconds to acquire client licenses. This process is repeated until at least one client license can be checked out (since ALTOS_QUEUE is set to 1).

Invoking Liberate

Users may do a quick version check of Liberate by executing this from the command line:

```
% 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"
```

System Libraries

Liberate is shipped enabled with dynamic linked system libraries. To verify **Liberate** is capable of running on your system, just try executing it. 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 Hspice as well as some Spectre and Eldo 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. The Tcl commands available for controlling Liberate are given in Chapter 3.

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 re-characterize an existing library to update the models or for a different PVT condition. To do this, use the **verbose** option to **write_template**. 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 re-characterization flow shown above will honor all of the arc conditions that pre-exist in a given library. The re-characterization flow is useful for updating an existing Liberate created library with a new PVT (process/temperature/voltage). 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, simply 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 re-characterize 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
```

Packet Mode

Liberate keeps all the cell 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. The alternative is to use only 64 bit machines or to break the library into smaller chunks of cells to use a network of 32 bit machines. Liberate can automate the later process using *packet mode*.

Liberate will initially estimate the memory required to characterize all of the cells that are loaded. If the estimate exceeds the limit specified by the parameter **bundle_mem_limit** the cells will be partitioned into separate packets. Each packet will use a separate Liberate run (without needing additional licenses), to characterize the cells in each packet. The output of each *packet* run will be a library database (**ldb**) as a directory rather than a single file**.**

The final model generation step (**write_library**, **write_verilog**, and so on.) should then be performed on a 64-bit machine 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.

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 to 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 easily 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.

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 actual 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   "/proj/tools/MMSIM/MMSIM131/tools.lnx86/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
```

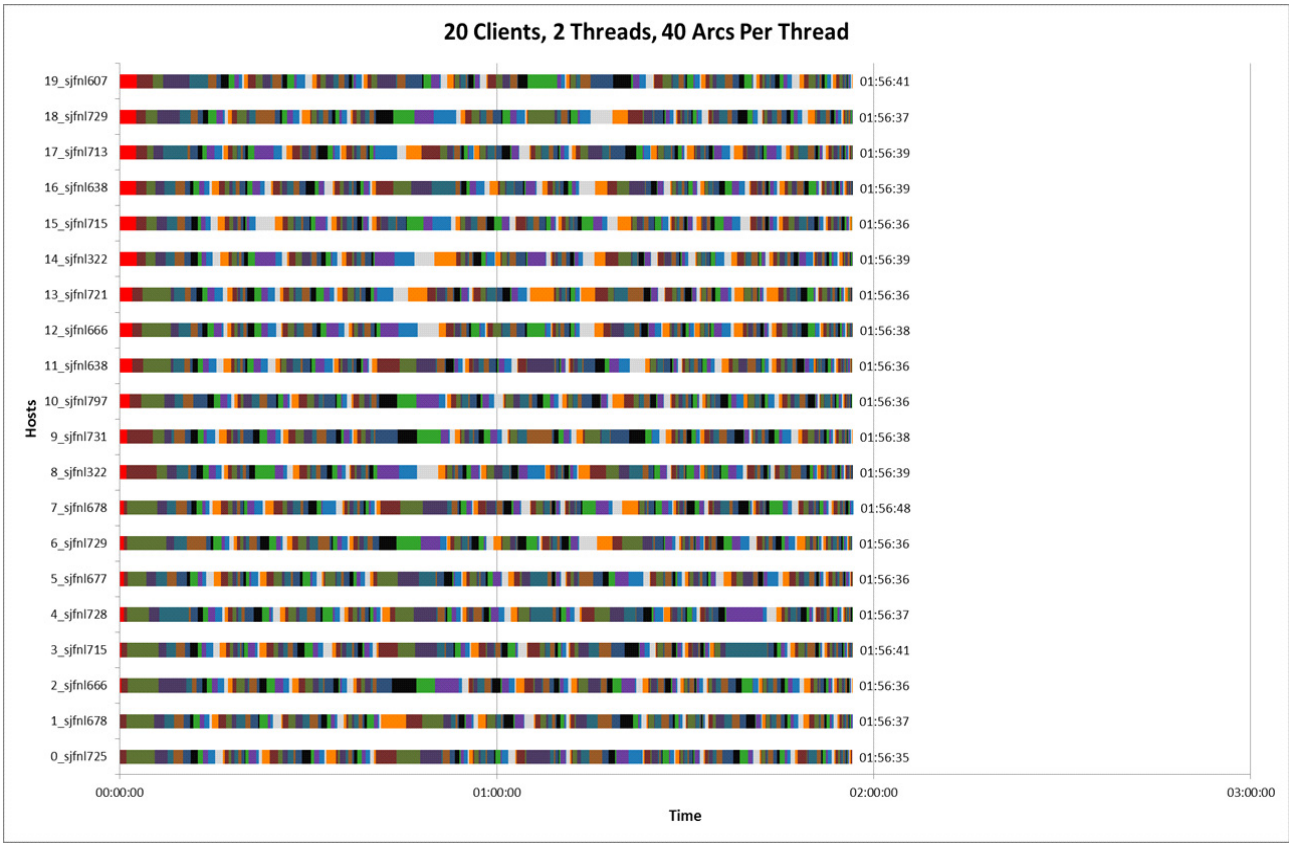
   The `logs2xlsx` utility automatically detects if the ldb directory contains cell-packet or arc-packet log files, parses them, and saves the data into the named `.xlsx` file. The `.xlsx` file can be opened with Microsoft Excel (2007 or later) or OpenOffice Calc on Linux machines. The `.xlsx` file has 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.

Licensing

The Arc Packet flow offers flexibility in how the `Liberate_Client` and SPICE licenses are checked out. The optimal method depends on the number of Liberate servers, Liberate clients, Spectre licenses, and demand from other users on the Spectre licenses. A block check out of licenses reduces the number of accesses to the Cadence license daemon that can prevent hitting a simultaneous access limit on extremely large installations. For flows not using Spectre, only the number of `Liberate_Client` licenses will be considered. To have the Liberate server manage the licenses for the Arc Packet flow, use the following command:

```
set_var  packet_arc_licensing_mode  wait_for_clients
```

Where:

`wait_for_clients` means that the Liberate server will check out a block of `Liberate_Client` and `Spectre_Char_Opt` licenses based on availability at the start of the job. If there are not enough licenses available, the server will continue to query the license DAEMON for additional licenses. (Recommended)

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 actual 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)

Th **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
```

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.

Enabling SKI

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 (32- or 64-bit). Set either both of the following environment variables or neither of them:

     ```
     setenv CDS_AUTO_64BIT all
     setenv ALTOS_64 1
     ```

   - Use the following Tcl settings:

     ```
     set_var ski_enable              1
     set_var ski_compatibility_mode  1
     set_var ski_clean_mode          1
     set_var extsim_model_include "<full_path>/model_include.sp"
     define_leafcell …
     ```

     Note: Ensure that the first line in the `model_include.sp` file is a comment. In addition, use the define_leafcell command for all instances (X*) and elements (M*) in the netlist file.

4

Liberate Commands

This chapter describes the Tcl commands that control library creation

All command arguments that are preceded with a dash (-) are optional except where explicitly indicated. All commands support the **help** option. When this option is specified with a command, a help message lists all currently available options for that command. In the help output, you might see options that are officially not supported. Only the options documented in this manual are supported. Therefore, ignore the undocumented command options.

add_lib_attribute

{attributes} Specify attribute(s) to add to a library.

add_cell_attribute

{cells} {attributes} Specify attribute(s) to add to cell(s) in a library.

add_pin_attribute

{cells} {pins} {attributes} Specify attribute(s) to add to the pin(s) of cell(s).

These commands provide the ability to add user-specified attributes to the ldb. Attributes may be applied to a library, a cell, or a pin, and may be simple or complex.

Example of adding attributes to a **library**:

```
add_lib_attribute  {
    wire_load ("5000") {
        resistance : 2.500000e-03;
        capacitance : 0.001000;
        area : 3.000000;
        fanout_length("1.0000000", "5.0000000");
    }
}
```

Example of adding attributes to a **cell**: (notice the use of wild cards for cell names)

```
add_cell_attribute {AN??D* AO3*} {
    test_cell () {
```

```
            ff (IQ,IQN) {
                clocked_on : "cp";
                next_state : "d";
                clear : "rn";
            }
        }
    }
```

Example of adding attributes to a **pin**: (wild cards used to match cell groups and pin names)

```
add_pin_attribute lv_buffer* a* {
    level_shifter_data_pin : true;
    input_signal_level : "dvdd2v8to1v8";
}
```

Sample usage:

```
# Read in ldb file
read_ldb test.ldb.gz

# Apply user data
add_lib_attribute {myAttribute1}

# Write .lib
write_library -ccs -overwrite test.lib
```

add_margin

-abs <value> Specify an absolute margin. Default 0.0 (no margin)

-cells {list} List of cells

-direction <rise | fall | both> Specify direction of data to add margin. Default: "both"

-pin {list} List of pins

-related {list} List of related pins

-rel <value> Specify a relative margin. Default 0.0 (0%)

-sensitivity_file <filename>
Specifies the sensitivity file to be loaded. The sensitivity file is created by Variety using the `write_variation` command with the `-format "sensitivity"` option. The -sensitivity_file option can be used only with one of following valid values of the `add_margin -type` option: `constraint`, `setup`, `hold`, `recovery`, and `removal`. Both the `add_margin -sensitivity_file` and `write_library -sensitivity_file` options can be used in the same run.

-type {list} Valid types include: `cap`, `constraint`, `delay`, `delay_ccs`, `hidden`, `hold`, `leakage`, `mpw`, `power`, `recovery`, `removal`, `retain`, `retain_ccs`, `retain_trans`, `setup`, `trans`, and `minimum_period`. Default: apply margin to all types.

-when "string" State dependent arc.

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

type specifies the type of data to be modified. Supported types are: **delay, trans, cap, constraint, leakage, power, hidden, setup, hold, recovery, removal** or **mpw**. If the type option is not specified, the requested margin will be applied to all data types.

constraint applies the same margin to all constraint types: **setup, hold, recovery, removal** and **mpw.**

abs (absolute) specifies the amount of margin to add in standard units. Default: 0.0 (no margin).

rel (relative) specifies a relative ratio amount of margin to add. If set to a positive number, the resulting library values becomes larger where larger is defined as more positive and smaller is more negative. If set to a negative number, the resulting library values is smaller if positive and larger if negative. Default `0.0` (0%). For example, 0.05 = 5% margin.

The types **power** and **hidden** are keywords used by `add_margin -type`. If you want to use these **types**, you may also use the `-cells` option. All other **types** can only be applied globally.

```
add_margin \
    -type {power|hidden} \
    -cells {celllists} \
    -pin {pin lists} \
    -related {related_pin list} \
    -when "when_condition" \
    -rel rel \
    -abs abs
```

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.

Examples:

```
read_ldb test.ldb.gz
write_library no_margin.lib

# Add 10% to power
add_margin -type power -rel 0.1

# Add 50ps to delay
add_margin -type delay -abs 50e-12
write_library margin.lib
read_ldb my.ldb
# Can also use read_library or char_library
add_margin -type setup -sigma 3 -sensitivity_file MySensitivity.dat { Cell1 }
write_library -filename my.lib slow
```

append_library

-allow_conflict	Allow libraries to be appended even if there are conflicts.
-filename <file_name>	Output file name. Default: <libname>.lib
-overwrite	Overwrite existing .lib file. Default: do not overwrite and create a unique file name.
-user_data <file_name>	Use this argument 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.
<libname>	Library name.

This command 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 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.

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.

Libraries not appended under these conditions

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.

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

Command Example

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

char_library

-auto_index	Automatically generate table indices
-auto_max_capacitance	Automatically generate `max_capacitance` attribute
-ccs	Characterize CCS delay data
-ccsn	Characterize CCS noise data
-ccsp	Characterize CCS power data
-cells {cell_names}	List of cell names (Default: *all cells*)
-client_postproc "string"	Command to execute after a client exits
-ecsm	Enables characterization of ECSM timing data. This option is on by default and is provided for script readability.
-ecsmn	Characterize ECSM noise data

-ecsmp	Characterize ECSM power data
-em	Requests electromigration characterization.
-exclude	Exclude **-cells** from characterization
-extsim <simulator_name>	Name of external simulator to use for characterization
-extsim_leakage <simulator_name>	Name of external simulator to use for leakage calculation
-ibis < 0 | 1 | 2 | typ | min | max >	Enables IBIS model characterization
-io	Characterize without "Inside View"
-si	Characterize old signal integrity data
-skip {delay | power | leakage | constraint | mpw | cin | setup | hold}	Skip characterization of selected categories. Default: Do not skip any data types
-skip_list {mycell {list of pins} {items to skip} <...> }	List of items to skip for a list of cells, or pin-wise for a list of cells.
-thread <number>	Number of different CPU threads to use
-trial	Generates a "dummy" lib with Verilog models. Only supports NLDM.
-user_arcs_only	Only characterize user specified arcs.

The **char_library** command performs library characterization. Each cell listed in a `define_cell` command will be characterized providing the SPICE subckt definition for that cell is defined in the netlists passed to the `read_spice` command. Only one `char_library` command is permitted per Liberate run.

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

auto_index instructs Liberate to automatically create the indices 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` indices set the `max_noise_width` parameter. Important: The `auto_index` option utilizes the inside_view algorithm and *cannot* be used with the -io option (since the -io option disables the inside_view). See Data Table Index Determination for detailed information on determining data table index values.

Important

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.

The `auto_index` option of `char_library` now requires `min_transition` and `min_output_cap` defined if Packet Mode is being used. If these are not defined, Liberate will generate an Error.
Note: In non-Packet Mode, `min_transition` and `min_output_cap` do not need to be defined, however, Liberate will output a Warning encouraging the user to set these variables.

auto_max_capacitance requests 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 actual 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. The **ccsn** argument will instruct Liberate to characterize composite current source noise (CCSN) data. The **ccsp** argument will instruct Liberate to characterize composite current source (CCSP) power data. The **ccsp** option is required when advanced power constructs are needed.

cells instructs Liberate to only characterize cells named in the list. By default, Liberate will characterize all the cells defined by `define_cell` commands**.** When used with the **exclude** argument, Liberate will exclude the cells in the list from characterization.

client_postproc 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.

ecsmn instructs Liberate to characterize the effective current source model noise (ECSMN) data.

ecsmp instructs Liberate to characterize the effective current source model power data. The `ecsmp` and **ccsp** options enable the same characterization so only one of these two options is really required.

em requests Liberate to characterize electromigration models. This is a standalone characterization that cannot be combined with other data formats such as `-ccs`, `-ccsn`, `-ccsp`, `-ecsm`, `-ecsmn`, and `-ecsmp`. This feature requires that Spectre to be used with APS (see the -extsim option of the char_library command and the variable extsim_cmd_option).

For more information, see the Electromigration Models section in the Liberate Details chpater.

extsim instructs Liberate to use an external SPICE simulator rather than Alspice**.** Alspice is the default built-in SPICE simulator. The license for the external simulator must be available. Currently, the following external simulators are supported

- HSPICE
- Spectre
- Eldo
- FineSim
- XA

The setting for **-extsim** must correspond to the setting for the extsim_cmd variable.

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_clients 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 specifies an external SPICE simulator to use for leakage calculations. (This is for leakage simulations only – all other simulations will use the default simulator, Alspice.) The user may also set the variable, extsim_model_include_leakage to specify a separate set of models for leakage calculations. If this is not set, Liberate will use the same set of models for all simulations.

IBIS enables the IBIS characterization for IO cells. The value tells Liberate which corner is to be characterized for IBIS format data.

io enables cell characterization without the use of "Inside View". This is useful for cells that contain analog or that are too large for digital vector analysis using Inside View. Using this

option turns off Liberate's automatic arc-determination and vector-generation. 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.

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` then noise immunity rejection curves will also be characterized.

skip can be used to disable characterization of specific categories of data. Supported categories are: `delay`, `power`, `leakage`, `constraint`, `mpw` and `cin`. A list containing multiple categories is supported. It is recommended that this option only be used to improve runtime 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 runtime for constraints.

Important

Skipping a category may 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 since this can lead to an incomplete library. The value `setup` will skip `setup`, `recovery`, `non_seq_setup`. The value `hold` will skip `hold`, `removal`, `non_seq_hold`.

skip_list allows the user to specify a list of data types to skip for a list of pins of a list of cells. If more granularity is needed, the user can specify a list-of-items to skip on a list-of-pins on a list-of-cells. The skip list may 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 * for a wildcard. Regular expressions such as "*1" are not supported at this time. Liberate will automatically recognize if the group after the cell is a list of skip items or a list of pins.

thread defines the maximum number of threads to use on the current machine. Even if the `thread` argument 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 instructs Liberate to run all of the preprocessing without running the actual 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 skips the automatic addition of arcs by the "inside view" of Liberate. This option requires that the user provide all required arcs using the define_arc command. This option is used so that the `write_template` verbose flow will more closely match the reference library structure.

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}

# Free up the Hspice license once each client exits
char_library -client_postproc "hspice -C -K"
```

check_delay_monotonicity

-adjust	Amount to adjust the delay or transition by when fixing non-monotonic data. Default: 0.001ps
-ecsm	Check ECSM waveforms
-exit	Exit if non-monotonic delay data (by load) is found
-fix	Fix non-monotonic transitions
-fix_ccs_delay	Fix any monotonicity problem caused by output load and/or transition for CCS delay.
-slew	Check monotonicity based on input slew
-transition	Check rise and fall transitions

The **check_delay_monotonicity** command 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`. The `ecsm` argument checks `ecsm_waveform` data while the `transition` argument checks *rise_transition* and *fall_transition* data. If the slew argument is specified then the checks are also performed with respect to input slew. That is, for a given load, all slews are checked. The `exit` argument will cause `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.

The **fix** argument will repair any rise/fall transition and any delay monotonicity problems (with respect to output load) by making the non-monotonic table entry equal to the previous entry plus 1 added to the least significant digit. The **exit** option will override the **fix** option.

The **fix_ccs_delay** argument controls whether the CCS delay data should be adjusted. This option should be used in combination with the **-fix** option whenever the library contains both NLDM and CCS timing data. This can help avoid mismatches between NLDM and CCS timing.

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
```

compare_ccs_nldm

-absolute_average	Report absolute average. Default: *report relative average*
-abstol <value>	Set the absolute tolerance for the CCS vs. NDLM error comparison. Default: 0.002 × time_unit
-**cells {cell_names}**	List of cell names. Default: *all cells*
-exclude	Exclude the list of cells from the comparison.
-format < txt | xls | htm >	Format the output for text, Excel, or HTML. Default: txt
-group <dirname>	Directory name to store cell comparisons for each cell group. Default: *all cells in a single report*
-gui <filename>	Generate a file for graphical comparison with **lcplot**
-lcplot	Display GUI output file using **lcplot**.

-nworst <number>	List the top <number> cells with outliers per data-type. Default: 5
-percent_max_diff	Report the percent error of the max difference. Default: *report the maximum percent difference*
-reltol <value>	Percentage tolerance for the CCS vs. NLDM error comparison. Default: 0.02
-report <filename>	Output comparison file name. Default: *<library_name>.cmp.txt*
-verbose	Report comparison of all data in the library, regardless of tolerances
<library_name>	Library name

The **compare_ccs_nldm** command compares the **CCS** data to the **NLDM** data in a single library and reports the differences that exceed the defined tolerances.

The **verbose** argument generates a report showing every comparison including those that did not exceed a tolerance. The output is written to the **report** filename, default <**library_lib>.cmp.txt**. An overall comparison summary is also written to the standard output. The **nworst** argument (default 5) 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. The **gui** filename defines the name of an intermediate file that can be used for graphical comparisons of data with the **lcplot** utility (see Chapter 5). The **lcplot** argument will invoke the **lcplot** utility for viewing the comparison results graphically. When using the **lcplot** argument, the **gui** argument is not required since a comparison data file called *<library_lib>.gui* will automatically be created.

The arguments **abstol** and **reltol** define absolute and relative tolerance limits for each comparison. Any comparison that exceeds both these tolerances is considered an outlier and will be reported. The default for **reltol** is 0.02 (2%). The default for **abstol** is 0.002 * *time_unit* (typically 2ns).

The **cells** option can be used to specify a list of cells to compare. By default, all cells will be compared. This option supports the use of a wildcard. If the **exclude** option is used, then the list of cells will be excluded from the comparison.

The **format** option can be used to specify the format for the output report. 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.

The **group** argument will request 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*

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

-absolute_average	Report absolute average. Default: *report relative average*
-abstol <value | {list}>	Set absolute difference tolerance. Default: 1e-3 × data_type_unit
-cells {cell_names}	List of cell names. Default: *all cells*
-comp_adjust_tristate_load <value>	Adjust tristate load of comparison library before comparing. Default: -1 (follow variable adjust_tristate_load)
-exact_match	Only compare arcs with identical logic ('*when*') conditions
-exclude	Exclude the list of cells from the comparison.
-format < txt | xls | htm >	Format the output for text, Excel, or HTML. Default: txt
-group <dirname>	Directory name to store cell comparisons for each cell group. Default: *all cells in a single report*
-gui <filename>	Generate a file for graphical comparison with **lcplot**
-index1_range <range>	*index1* range to use for comparison. Default: *use all indices*
-index2_range <range>	*index2* range to use for comparison. Default: *use all indices*
-lcplot	Display GUI output file using **lcplot**.
-lib <abs | rel>	Request a Liberty formatted report with absolute or relative data differences.
-multiple_matches	Report comparison of all arcs that have functional overlap with a reference arc
-no_interpolation	Skip comparisons on tables with different indices. No interpolation between index points will be performed.
-nworst <number>	List the top <number> cells with outliers per data type, default 5
-padding	Pad delay, transitions and constraints by ½ input slew. Pad power by an additional $\frac{1}{2}CV^2$
-padding_index	Select same, mid, or end slew index for padding.
-percent_max_diff	Report the percent error of the max difference. Default: *report max percent difference*

-ref_adjust_tristate_load <value>	Adjust tristate load of reference library before comparing. Default: -1 (follow variable adjust_tristate_load)
-reltol <value|{list}>	Set percentage difference tolerance Default: 0.01
-report <filename>	Output report file name. Default: *<comp_lib>.cmp.txt*
-skip {list}	List of data-comparison types to skip. Default: *none*
-type {list}	List of data-comparison types to include. Default: *all*
-unmatched	Report unmatched data entries
-verbose	Report all comparisons regardless of tolerances
<ref_lib>	Reference library
<comp_lib>	Comparison library

The **compare_library** command compares the **comp_lib** library against the **ref_lib** reference library and reports the differences that exceed the defined tolerances. The report includes the comparison of SI, ECSM and CCS, and CCSN data in addition to the comparison of attributes, capacitance, leakage, delays, transitions, power and timing constraints. For CCS, the current waveforms are converted to voltage waveforms and the comparisons performed using delay and slew thresholds rather than for each current measurement. If the table indices in the comparison library are different from the reference library, bi-linear interpolation will be used prior to performing the comparison. For CCSN, the following data types are supported: ccsn_dc, ccsn_vout and miller_cap (propagation tables are not yet implemented). For ccsn_dc and ecsm, 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 etc.). If the reference value is zero and the comparison value is not zero then 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.

When comparing libraries, the data entries must have equivalent conditions. Two entries are deemed equivalent if they have the same or overlapping logic conditions, related pins and data-type. 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** argument. To compare entries only when there is an exact match in the *when* conditions, use the **exact_match** argument. 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 that all the pin names must match. Specify the **multiple_matches** argument 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 e.g. :

```
| when : !M1 Vs (!(M1) * !(M2)) (1 of 2), Timing : combinational
```

Note: The **exact_match** argument will override the **multiple_match** argument.

The **verbose** argument generates a report showing every comparison, including those that did not exceed a tolerance. The output is written to the **report** filename, default *<**comp_lib**>**.cmp.txt***. An overall comparison summary is also written to the standard output. The **nworst** argument (default 5) 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. The **gui** filename defines the name of an intermediate file that can be used for graphical comparisons of data with the **lcplot** utility (see Chapter 5). The **lcplot** argument will invoke the **lcplot** utility for viewing the comparison results graphically. When using the **lcplot** argument the **gui** argument is not required, as a comparison data file called *<comp_lib>.gui* will automatically be created.

The **padding** argument 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 ½CV2 (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. The **padding_index** option can be used to specify which slew index to use when adding padding. Legal values are: same, mid, and end. The default is to use the same slew index as for delay.

The **no_interpolation** argument can be used to disable the comparison of data groups that have mismatched numbers of indices. By default, if the number of index values is different then the comparison values will be interpolated.

The arguments **abstol** and **reltol** 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 argument only has a single value, then the type for that value is assumed to be *all*. The **abstol** value should be given standard units (not library units) i.e. "*delay 5e-12"* to set the **abstol** for delay to 5ps. The default of **reltol** is 0.01 (1%). The default of **abstol** is 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.

The **cells** option can be used to specify a list of cells to compare. By default, all cells will be compared. This option supports the use of a wildcard. If the **exclude** option is used, then the list of cells will be excluded from the comparison.

The **format** option can be used to specify the format for the output report. The default (**txt**) is a standard textual format. The **xls** value can be used to produce an output format that is more

suitable for import into Microsoft Excel®. The htm value can be used to request output in HTML 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.

comp_adjust_tristate_load and **ref_adjust_tristate_load** control if pin capacitance should be added to the load indices on tristate pins. Accepted values are:

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

The **group** argument will request 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*

The **type** argument specifies a list of data comparison types to compare. The default is *all.* The **skip** argument specifies a list of data comparison types to skip. The default is *none* (don't skip any comparison types).

Valid comparison types are:

```
attributes, cap, ccs, ccs_cap, ccs_retain, ccsn_dc, ccsn_prop, ccsn_vout, ccsp,
ccsp_cap, ccsp_dc, ccsp_lc, ccsp_res, clear, delay, delay_variation, ecsm,
ecsm_cap, ecsm_cap_variation, ecsm_variation, hidden_power, hold, hyper,
in_cap, leakage, max_cap, max_trans, miller_cap, mpw, noise, nonseq_hold,
nonseq_setup, power, preset, recovery, removal, retain, retain_slw, setup, siv,
three_state, three_state_disable, three_state_enable, time_const, trans,
trans_variation, tristate
```

In addition, a small collection of "macro-types" are available. These are simply 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 **skip** and **type** arguments separate dynamic power from hidden power. Specifying **-skip {power}** will skip dynamic power, specifying **-skip {hidden_power}** will skip hidden power, and **-skip {power hidden_power}** will skip both.

The **index1_range** and **index2_range** arguments can be used to limit the comparison to a range of *index1* or *index2* values. The range is either two values separated by a "-" e.g. "1-3" to compare the first three indices or a single value e.g. "2" to compare the second index only.

When comparing two libraries that have different index values, slew thresholds and units, the values in the **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 indices were switched
+ : both the ref_lib and comp_lib values were padded.
```

When comparing libraries that have different *when* conditions, the data groups that have overlapping conditions will be compared. If the number of indices (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 indices 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.

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*.

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 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: If the index_1 and index_2 values are not exactly matching, then the right side of the row will show a ^, ~, / or # to indicate either extrapolation or interpolation occurred. See the Legend 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 %: Sqrt(Abs(((Sum of Diff%^2)/ #Entries) - (Avg Diff% ^2)))
- Max Diff: The largest Diff value
- Max Diff%: The Diff% related to the Max Diff.
- 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.

Sample Summary

```
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 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

copy_arc

-from_cell <name>	Cell to copy from.
-from_pin <name>	Pin to copy from.
-from_related <name>	related_pin to copy from. Default: copy arcs regardless of related pin

-from_timing_sense <value> Timing sense to copy from. (Example: positive_unate) Default "" (Copy regardless of timing sense)

-from_timing_types {list} Timing types to copy. Default: copy regardless of timing type

-from_when <value> *When* condition of arc to copy from. Default: "*" (all conditions)

-margin {list} Amount of absolute margin to add after copying the arc. Default 0.0 (no margin.)

-method < replace | append | augment > Copy method to use "replace, append, or augment". Default is to replace existing matching arc.

-rel_margin {list} Relative amount of margin to add after copying the arc. Default: 0.0 (0%)

-to_cell <name> Cell to copy to. Default: -from_cell

-to_pin <name> Pin to copy to. Default: -from_pin

-to_related <name> related_pin to copy to. Default: -from_related

-to_timing_sense <value> Timing sense to copy to. Default: "" (use -from_timing_sense)

-to_timing_types {list} Timing types to copy to. Default: use -from_timing_types

-to_when <value> *When* condition to use after copying the arc. Default: use the -from_when condition

-type {list} Specifies the type of data to be copied. Default: copy all data types.

The **copy_arc** command is used to copy one or more arcs from one cell to another (or from one arc to another in the same cell).

from_cell, **from_pin**, **from_related**: these options specify the cell, pin, and relate_pin for the arc to be copied.

to_cell, **to_pin**, **to_related**: these options specify the cell, pin, and related_pin where the arc information will be copied to.

margin specifies the absolute margin to be added to the arc and accepts a list of values. 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.)

rel_margin specifies the relative margin to be added to the arc and accepts a list of values. rel_margin can specify a single value, or a list of values (similar to margin.)

type specifies the type of data to be copied. Supported types are: ccs, ccs_retain, constraint, delay, hold, mpw, power, recovery, removal, retain, setup, and timing.

define_arc

-attribute {list} List of user-defined attribute/value pairs

-constraint <value> Logic constraint *(Deprecated. Use -logic_condition)*

-delay_threshold {in_rise in_fall out_rise out_fall}
List of four delay threshold values

-dependent_load {<pin load_value>...} List of pin-load pairs to add to dependent side pins. (See set_dependent_load)

-dual_dir <U | D | B> Switching direction of dual pin (Up, Down or Both) used to set load direction.

-dual_pin <name> Name of the other pin in a differential output pair

-dual_related <name> Name of the related differential input

-extsim_deck_header Includes arbitrary spice syntax commands in the extsim deck.

-ic <"ic list"> Initial conditions for each pin in the pinlist

-ignore Flag to indicate that this arc should be ignored.

-load_dir <U | D | B> Output load direction (pullUp, pullDown or Both)

-logic_condition <value> Logic condition

-margin Allows a user-specified backoff margin

-metric {list} Specifies a list of one or more metric types for measuring timing constraints.

-metric_thresh {list} When combined with `-metric` option, overwrites the value set by `set_constraint_critera`, or by the relevant variable for this arc. This list must have one value corresponding to each criteria in the metric list:

-pin {pins} List of pin names (REQUIRED)

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

-pin_gnd {pin voltage …} Arc specific input pin gnd by pin/voltage pairs.

-pin_load <template> Predefined pin loading. (See define_pin_load)

-pin_load_dir {pin template dir}Applies specific pin_loads to specific output pins.

-pin_probe {pins} List of pin monitor node names

-pin_probe_dir {pins} List of pin monitor node directions [R|F]

-pin_probe_threshold {pins} List of pin monitor node thresholds.

-pin_vdd {pin voltage …} Arc specific input pin vdd by pin/voltage pairs.

-pinlist {list of pins} List of pins corresponding to vector

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

-prevector_pinlist {list} User specified pin list for pre-vectors.

-prevector "<vector... >" User specified initialization vectors (Default: no prevectors)

-probe < {names} | altos_internal >
List of names of nodes to monitor for sequential cells constraint characterization or keyword.

-probe_dir <R | F> Monitor node direction.

-related_pin {pins} List of related pin names

-related_pin_dir <R | F> Transition direction of related pin(s)

-related_probe {pins} List of related monitor node names.

-related_probe_dir {R | F} List of related_probe monitor node directions [R|F]

-related_probe_threshold {pins} List of related monitor node thresholds.

-sdf_cond <"function"> Logic conditions of side inputs for sdf_cond

-slew_threshold { lower_rise upper_rise lower_fall upper_fall }
List of four slew threshold values

-type < async | combinational | disable | edge | enable | hidden | hold | mpw | non_seq_hold | non_seq_setup | power | recovery | removal | setup >
Type of arc (Default: *combinational*)

-value <value> Overrides characterized values (Default: use characterized values)

-value_trans <value> Overrides values in the transition table. (Default: use characterized values.)

-vector <"stimulus"> Vector stimulus used to simulate this arc, each bit can be one of: *R | F | X | 1 | 0*

-when <"function"> Logic conditions of side inputs

{cell_names} List of cells

The `define_arc` command specifies a user-defined arc to override Liberate's automatic arc determination. An "arc" represents library data between a given pin and a related pin. Typically this command is only required for the characterization of complex I/O cells. When 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`.

`define_arc` can be applied to a single cell or a list of cell names. The templates to use for each arc will default 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` 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.

attribute accepts a list of attributes/value pairs. These attributes and their values will be added to the .lib for this specific arc.

delay_threshold and **slew_threshold** arguments define a list of delay and slew percentage measurement points (a ratio of VDD normalized to between 0 and 1) for the arc. Each argument consists of a list of four values.

For `delay_threshold` these values represent the measurement thresholds:

```
input_rise_delay, input_fall_delay, output_rise_delay,
output_fall_delay
```

in that exact order. If not specified, then all delays are measured at the values defined by the `delay_*_*` parameters.

For `slew_threshold` the values in the list represent the measurement thresholds:

```
lower_rise_slew, upper_rise_slew, lower_fall_slew,
upper_fall_slew
```

in that exact order. If not specified, then all slews are measured at the values defined by the `measure_slew_*` parameters.

dependent_load provides user control over the load applied to side outputs that impact the arc. These are specified as a list of pin-load pairs, i.e: {pin1 load1 pin2 load2 ...}. Dependant loads in the `define_arc` command will supersede those specified by the `set_dependent_load` command.

dual_pin specifies the other pin in a pair of differential output pins. The `dual_dir` argument is the equivalent of the `load_dir` argument as it defines the direction of the load circuitry to apply to the `dual_pin` of this `define_arc` command. The `dual_dir` can be `U` (up), `D` (down) or `B` (both).

dual_related argument specifies 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 argument is intended to be used when an external simulator is used (refer to the `-extsim` argument of the **char_library** command). It is a local arc specific version of the variable `extsim_deck_header`. As Liberate does not parse the string specified by this argument, 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 defines initial conditions to be applied during simulation. There should be one voltage value for each pin in the pinlist.

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.

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

The `when`, `logic_condition` and `vector` option must be consistent or the define_arc command will be ignored. That is, 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 is used for setting constraint criteria for timing constraint arcs and may contain a list that includes the following keywords corresponding to Liberate variables:

delay | 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 may not 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 `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 option `metric_thresh` (below); or the threshold set with the `set_constraint_criteria` command.
- 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` variable; the threshold set with option `metric_thresh` (below); 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` 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 }
```

metric_thresh takes a list of threshold values and is used on combination with `metric` to override values specified by `set_constraint_critera`, or by the corresponding global variable:

slew: `constraint_slew_degrade`

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

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
```

pin_gnd and **pin_vdd** options can be used to specify arc specific input voltages. These commands accept a list of pin/voltage pairs. These options can be used without the `vector` option.

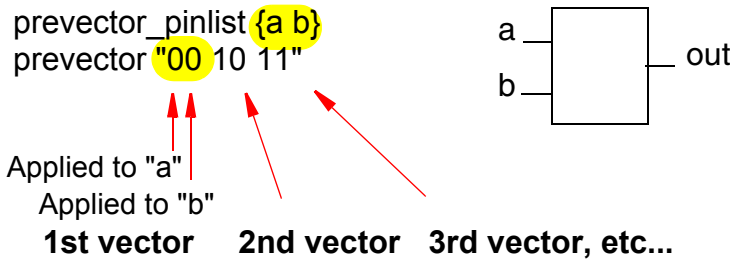
pin_load argument can be used to specify additional circuitry to be applied to all the destination `pins` of the `define_arc` command. The `pin_load` argument refers to a template that defines the loading circuitry to be placed prior to the loading capacitance for the pin. The loading template must be pre-defined using the define_pin_load command. The additional circuitry can include pullup and pulldown resistances and series resistance. The `load_dir` argument defines whether one or both of the pullup or pulldown resistances should be applied to this arc. Setting it to `U` (up) will include the pullup resistance, setting it to `D` (down) will include the pulldown resistance, while setting it to `B` (both) means both pullup and pulldown resistors are included.

pin_load_dir applies specific loads to specific output pins, providing more granularity of control than `pin_load`. This option takes a triplet of arguments: `pin_name, load_template, direction,` and may be specified as a list of triplets. `load_template` is the name of the loading configuration defined with `define_pin_load`, and direction is specified with `U` (pullup resistance), `D` (pulldown resistance), or `B` (both pullup and pulldown resistances).
Note: `pin_load_dir` overrides any `pin_load` that may be present. Example:

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

prevector specifies an arbitrary simulation stimulus to be applied before the actual 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 variables: 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:



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 Inside View 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)

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
```

probe is used for timing constraints and defines the node(s) to monitor when determining the constraint. It can be an external pin such as the `Q` pin in a flip-flop or an internal node name. 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 then the pin defined by the constraint_output_pin parameter is probed.

probe_dir is used with the `probe` option and specifies the direction that the probe nodes will be transitioning. Legal values are: `R|F`. One direction can be provided for each probe node.

related_pin is a list of related pins (typically input pins for combinational arcs, clock pins for timing constraint arcs) while the `pin` argument is a list of destination pins for the arc (typically output pins for combinational arcs, input pins for timing constraint or hidden power arcs).

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

sdf_cond defines the logic conditions of the other pins of the cell to enable this arc using SDF compatible syntax.

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

type 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, 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.

value overrides the results of characterization and forces a value for all entries into the data table for the specified arc. `value` takes either a list or a single point; if a single point is given, Liberate interprets this as a list with a length of one. For a time-based arcs, the value is in seconds. (i.e.: 5e-9 = 5ns.)

value_trans is similar to `value` except that it applies to transition tables. This option allows the user to provide transition values to items into the transition table. If a single value is provided, then the table will get a scalar type of data. If a list of values is provided, then there must be one value for each table entry (ex: a 7x7 table would require 49 values).

vector defines the stimulus to simulate this arc. It is defined as a string of bits where 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 to the pin list order defined by the define_cell `pinlist` argument or the `define_arc pinlist` argument. White space is permitted in the vector for readability. Note if a side input is specified as `X` but the `when` condition requires it to be logic high (`1`) then the side input will be set to `1`. In addition, all stimulus vectors that satisfy the `when` condition will be enumerated and simulated.

Caution: The user should not define a vector that drives a node such that it cannot be resolved (i.e.: 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 will ignore the define_arc command that caused this condition, and output a warning message.

when 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.

This command must be used before **char_library.**

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
```

define_bundle_pins

-use_pin <pin_name>	Specifies the "representative" pin
<cellName>	The name of the cell.
<bundle_pin>	The name of the bundle.
{pins}	List of pins in the bundle

Instead of including each of the input and/or output pins, the pins are treated as if they were from equivalent latches or flip-flops in a bank. For example, pins SE1 and SE2 are from separate flip-flops that are identical, so 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.

use_pin 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 then the command must be used *before* **char_library**.
> If this option is not used, then define_bundle_pins may be used after char_library, but before **write_library**.

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

-by <integer>	Specifies the bus increment (Default:1)
-dont_overwrite	Don't overwrite previous bus definition (Default: overwrite)
-from <integer>	Specifies where the bus starts (Default: 0).
-to <integer>	Specifies where the bus ends (Default: 0)
-use_bit	Specifies the bus bit to use to characterize the bus (Default: the value of the -from option)
cellName	Specifies the cell name (Default: "cell name").
bus_name	Specifies the bus name (Default: "bus name").

The **define_bus** command is used to group bus bits into a bus in the output library. It is not required if buses are explicitly defined in **define_cell.**

The **bus_name** should be the name of the pins minus the bus index.

The **–from** argument can be higher or lower than the **–to** argument, but not equal. These arguments define the bus start and ending range.

The **-by** argument defines the bus bit increment.

The **-use_bit** option specifies the bus bit whose characterized data will be applied to the entire bus.

The **-dont_overwrite** option specifies that multiple define_bus commands for the same bus will be merged.

This command must be used before **write_library** and can also be used after **char_library** or **read_ldb.**

define_cell

-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
-delay <name>	Name of template for delay tables
-harness <load_subckt_name>	Name of subckt containing custom load
-ignore_input_for_autocap {pin_names}	List of input pin names
-ignore_pin_for_ccsn {pin_names}	List of input pin names not to have ccsn data.
-input {pin_names}	List of input pin names

-internal {pin names}	List of internal pin names **Note**: This argument is available only in Liberate AMS.
-internal_supply {supply_names}	List of switched supply pin names
-mpw <name>	Name of template for mpw tables
-output {pin_names}	List of output pin names
-pinlist {pin_names}	Pin name order, used by define_arc
-power <name>	Name of template for power tables
-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
-type <normal | mega | io>	The type of cell (Default: *normal*) Note: In **define_cell**, `-type uda` has been renamed to `io`. Both `io` and `uda` are accepted and executes the same code. This means that they are synonymous.
-user_arcs_only	Skips the automatic addition of arcs by the "inside view" of Liberate. This argument 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 argument is often used with the write_template-verbose command to ensure that the new library exactly matches the reference library structure.
-when <"function">	User-specified cell level logic constraints
{cell_names}	List of cell names to be characterized

The **define_cell** command 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.

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 pre-processing and characterization time for large cells. Note that mega mode is currently in beta phase, as the primary focus is on timing and not power. Using **uda** disables Liberate's automatic vector generation and characterizes only user-defined arcs (**uda**).

The **input/bidi/output/clock/async** arguments 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 may not have **clock** or **async** pins so 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 then they must be defined uniquely with separate **define_cell** commands.

The **pinlist** argument is used to define the pin-order and is used by the **vector** argument of **define_arc** when specifying a user-defined timing arc. The pin list can contain internal pins as well as input, inout and output pins.

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

The **delay** argument 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.

The **power** argument 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.

The **constraint** argument 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 pre-defined using the **define_template** command.

The **mpw** argument 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.

The **si_immunity** argument defines the template to use 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** argument to **char_library** must also be set for signal integrity characterization.

The **harness** argument has been added to allow the user to specify an arbitrary spice format circuit to "wrap" around the cell that is being characterized. The user provides a subcircuit containing an arbitrary input circuit and/or an arbitrary output load. The harness sub-circuit must have been loaded with **read_spice**. The harness sub-circuit 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

The **internal** argument is used to define pins that are internal to the cell, that is, they are not part of the port list of the top level subckt definition for the cell. Such pins must be defined when they are used as `-pin` or `-rel pins` argument of the `define_arc` command.

The **internal_supply** argument 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 powerdown function attribute. All internal_supply pins must also be identified as a supply using the set_gnd / set_vdd 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 option **scan_scale_power_factor**. 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.

The **user_arcs_only** argument skips the automatic addition of arcs by the "inside view" of Liberate. This option requires that the user provide all required arcs using the **define_arc** command. This option is used so that the `write_template` verbose flow more closely matches the reference library structure.

The **when** argument enables specification of user defined logic constraints using the Liberty format *when* syntax, constraining Liberate's automatic vector generation for this 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.

The **ignore_input_for_auto_cap** argument accepts a list of input/bidi pins. Any timing arcs which originate at pins specified in this list will not be considered for **auto_index** and **auto_max_capacitance** calculations.

The **ignore_pin_for_ccsn** argument accepts a list of input/bidi pins. No CCSN data will be characterized for the specified 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

-echo Echo each generated **define_leakage** command into the log file.

-active_pinlist {list} List of pin names to expand into different states.

-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_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.

Use this command to create 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”.

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 x} na2
define_cell_leakage \
        -echo \
        -active_pinlist "b"
        -static_pinlist “a” \
        -static_vector “1” \
        na2
```

The following commands will be echo’d into the log file and executed:

```
define_leakage  -when "!b" -vector "10x" na3
define_leakage  -when "b" -vector "11x" na3
```

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 echo’d into the log file and executed:

```
define_leakage  -when "!d*!clk" -prevector_pinlist "clk" -prevector "0 1 0" -vector
"001x" dff
define_leakage  -when "!d*clk" -prevector_pinlist "clk" -prevector "1 0 1" -vector
"011x" dff
define_leakage  -when "d*!clk" -prevector_pinlist "clk" -prevector "0 1 0" -vector
"101x" dff
define_leakage  -when "d*clk" -prevector_pinlist "clk" -prevector "1 0 1" -vector
"111x" dff
```

define_duplicate_pins

<cellname >	The cell name to apply the duplication to.
<pin>	The pin to be duplicated.
{ duplicate_pins }	List of pin names to be duplicated.

The **define_duplicate_pins** command 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 will copy 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. Example:

```
define_duplicate_pin mux A { B C D E F G H I }
```

This command also supports duplicating a complete bus. 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 doesn't need to exist - it can be created on the fly during write_library.
- Duplicating bundles is not supported.

define_group

<group >	Group name
<"description">	Text describing the group
{cell_names }	List of cells belonging to this group

The **define_group** command defines a cell group and a text description for that group. This information is used by the datasheet generator (**write_datasheet**) to group cells with similar functionality. The define group commands 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 will be grouped based on their footprint. If no footprint information is available then all cells will belong to their own unique group.

The grouping information is accessible via two API functions **ALAPI_cellgroups** and **ALAPI_description**. The function **ALAPI_cellgroups** will generate a list of groups from the library while **ALAPI_description** will return the text description of a given cell. For more information on the API functions, see the ALAPI manual.

Example:

```
# Define a group of OR gates
define_group OR2_gates "2 input OR" {OR2_1 OR2_2 OR2_4}
```

define_index

-index_1 {indices}	List indices to use as index_1
-index_2 {indices}	List indices to use as index_2
-pin {pins}	List of pin names (REQUIRED). Accepts wildcards.
-related_pin {pins}	List of related pin names. Accepts wildcards. **Note**: The `-related_pin` 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.
-type {data_types}	List of data types *constraint, delay*, *mpw*, *power* or *si_immunity*
-when <"function">	Logic state of side inputs.
{cell_names}	List of cell names. Accepts wildcards.

The **define_index** command overrides the indices specified in the templates referenced by **define_cell,** or created using the **auto_index** argument 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.

pin, **related_pin** and at least one of **index_1** or **index_2** must be specified.
Note: pin and related pin accept the asterisk "*" wildcard. Example:

```
define_index -pin D*  # All pins beginning with "D"
define_index -pin *  # All pins
```

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 your char_library command used auto_index, define_index is applied *after* auto_index completes. If define_index is successfull (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

-direction < rise | fall > Direction of the input waveform. (REQUIRED)

-gnd_val <voltage> gnd voltage value of the associated input pin(s) in volts. (REQUIRED)

-pinlist {cell pin <cell pin> ...} List of pins to apply the waveforms. Specified in cell/pin pairs.

-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> The slew from the index in ldb/lib units. (REQUIRED)

-vdd_val <voltage> vdd voltage value of the associated input pin(s) in volts. (REQUIRED)

Use this command to specify 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. This command must be used before **char_library**.

direction is used to specify the logical direction of a given input waveform. Valid direction values are `rise` and `fall`.

vdd and **gnd** voltages specify the input voltage full rail swing for the cell/pin.

pinlist is a list of cell-pin pairs. 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.

slew_index specifies the index transition value that the pwl waveform is linked with.

Example:

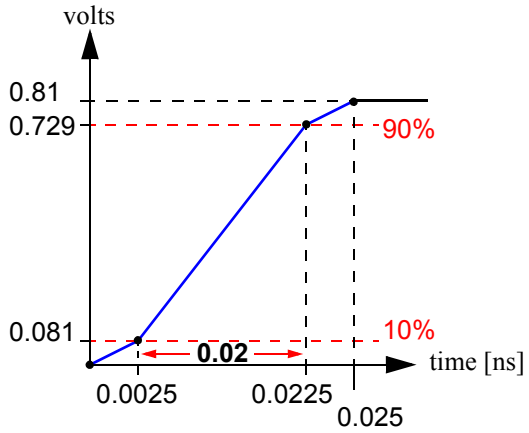
This sample shows a template file that has slew thresholds set at 10% and 90%, and slew index values set at 0.02 and 0.555:

```
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_*` variables 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:

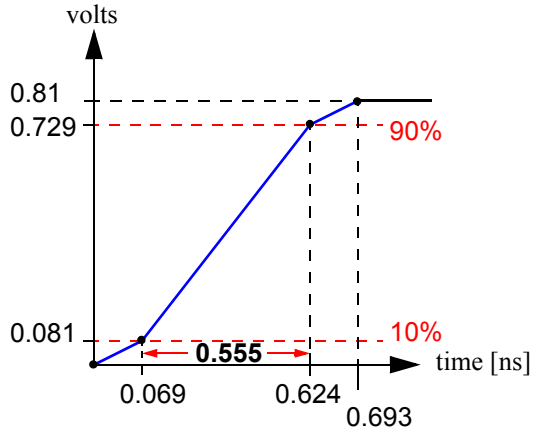
```
slew_derate = (measure_slew_upper_rise - measure_slew_lower_rise)/(slew_upper_rise
- slew_lower_rise)."
```

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

-area "parameter_name" Name of area diode parameter (default 'area')

-element Enables circuit element based leaf cells. Model name(s) must be specified.

-extsim_model Allows for partial include and partial read_spice

-length "parameter_name" Name of the Length mos parameter (default 'l')

-multiple "parameter_name" Name of Multiple mos parameter (default 'm')

-nfin "parameter_name" nfin parameter for FinFets. Default: 'NFIN'

-pin_position {list of pin positions}
Pin positions usually start from 0 <drain gate source [bulk(s)]> | <terminal_p terminal_n [bulks]>(REQUIRED)

-pj "parameter_name" Name of pj diode parameter (default 'pj')

-scale "value" mos param scale factor (default '1.0')

-type <nmos | pmos | diode | r | c | nmos_stk | pmos_stk | npn | pnp | black_box >
Type of cell

-width "parameter_name" Name of Width mos parameter (default 'w')

{leaf_cell_names} List of leaf cell names

This command 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` variable to enable external simulation (see `-extsim` argument 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.

area provides the name of the diode area parameter in the cell. The default name is: 'area'.

element enables circuit element based leaf cells. Without this option, the define_leafcell applies only to instances in the netlist. An instance in Spice format has a name beginning with "X". When this option is turned on, the define_leafcell is applied only to 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), etc. If the netlist has instances (preceding "X") , then this option should not be used.

extsim_model loads the model files for the leafcell(s). If this argument is used, the leafcell 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 leafcells does not have the `extsim_model` argument nor the `extsim_model_include` present, the tool will output an error requesting use of the `extsim_model_include` variable/parameter and quits.

length provides the name of the mos Length parameter in the cell. The default name 'l'.

multiple provides the name of the mos Multiple parameter The default name is 'm'.

nfin provides the name of the parameter in the netlist instance (or element) that maps to the number of fingers of a FinFet process.

pin_position maps the pins of the leafcell in the netlist to the pins of the corresponding subckt or model in the model file. There should only be one number for each pin in the leafcell. The first pin is designated by 0.

pj provides the name of the diode pj parameter in the cell. The default name is: 'pj'.

scale provides the scale in the cell. The default is: '1.0'. This scale factor is used only by the Liberate "Inside View" to determine device sizes, and is not applied to the device sizes in the simulation netlist.

type 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 5 pin stacked NMOS/PMOS transistors. For 7 pin stacked MOS, the extra 2 pins are internal pins. Note that the `pin_position` for stacked MOS is: 'd g1 g2 s b'.

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.

width provides the name of the mos width parameter in the cell. The default name is: 'w'.

If **define_leafcell** commands in the char script have the `extsim_model` argument included, there are two ways to load model files for these leafcells – by using the `extsim_model_include` variable or using the `extsim_deck_header` variable. All other device models may be loaded by using `read_spice`.

This command must be used before `read_spice`. If the `extsim_model` argument is used with this command, it must be used before `char_library` and `write_library` as well.

Example 1:

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

```
        PCH_map
```

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
```

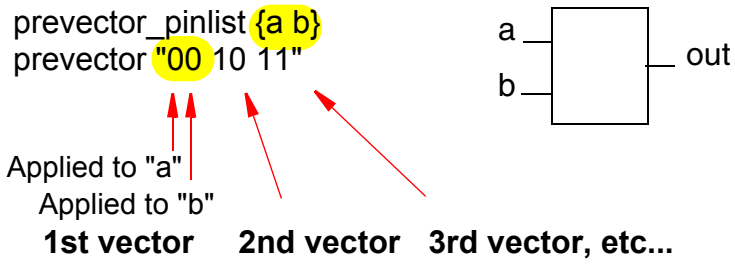
define_leakage

-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 argument is intended to be used when an external simulator is used (refer to the `-extsim` argument of the **char_library** command). It is a local arc specific version of the variable `extsim_deck_header`. As Liberate does not parse the string specified by this argument, 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 ..`
-prevector <**"vector"**>	User-specified initialization vectors (Default: no prevectors)
-prevector_pinlist {list}	User-specified pin list for pre-vectors.
-when <"function">	User-specified logic conditions (REQUIRED)
-vector <"stimulus">	Vector stimulus used to simulate the leakage, each bit can be one of X | 1 | 0
{cell_names}	List of cell names

The **define_leakage** command defines the logic conditions to use for calculating leakage power for the given cells. Using this command turns off Liberate's automatic determination of leakage conditions for the listed cells.

prevector specifies an arbitrary simulation stimulus to be applied before the actual 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 variables: prevector_period, and prevector_slew.

This option must be used in conjunction with `prevector_pinlist`. The following illustrates how prevectors are specified with `prevector_pinlist`:



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 Inside View 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)

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_leakage \
  -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_leakage \
       -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_leakage \
```

```
-type min_pulse_width \
-prevector_pinlist {D CK Q} \
-prevector {100 111 011} \
-vector {0FF} \
-related_pin CK \
-pin CK \
DFF
```

The ***-when*** argument specifies the logic conditions using the Liberty *when* format syntax. This command should be used when characterizing I/O cells that do not use automatic vector generation (**char_library –io**).

The -**vector** argument allows for partial when conditions for leakage in the output library, while having secondary pins set to a mix of 0s and 1s.

Note: Enhanced leakage characterization is also achievable by the tool honoring the **–when** condition specified in the **define_cell** command.

This command must be used before read_spice, char_library, and write_library.

Examples:

```
# 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
```

define_map

<map_filename> Name of the map file

The **define_map** command defines a file for mapping cell names prior to writing out the template, library, Verilog, VITAL, or datasheet files. It can also be used to map cell names when doing a library comparison using the **compare_library** command. It also changes the 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**.

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

-cells { }	List of cells.
-pinlist { }	List of pins.
< limit >	The max_capacitance attribute limit value.(in Farads)

This command is used to set a pin-specific maximum capacitance attribute limit. 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

<value>	Maximum allowable capacitance (in Farads).
{<cell> <pin> … }	List of cell pin pairs

This command is used to set a pin-specific maximum capacitance. This command only has effect when using the **auto_index** option to **char_library**. The values set by **define_max_capacitance_limit** override the calculated max_capacitance when the limit is exceeded. Multiple **define_max_capacitance_limit** commands can be specified. A wildcard * is supported 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

<value>	Maximum allowable transition time (in seconds).
{<cell> <pin> ...}	List of cell/pin pairs

This command is used to set a pin-specific maximum transition. This command only has effect when using the **auto_index** option to **char_library**. The values set by **define_max_transition** override the global value set by the **max_transition** parameter. Multiple **define_max_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.

Caution: 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 (i.e.: char.tcl) should have a read_vdb -> char_library -auto_index flow. If both scripts use -auto_index then 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 0.5e-9 {DFFX1 CK LTX1 LCK}
define_max_transition 0.75e-9 {GATER CLKIN * CLK}
char_library -auto_index
```

define_min_transition

<value>	Minimum allowable transition time (in seconds).
{<cell> <pin> ...}	List of cell/pin pairs.

This command is used to set a pin-specific minimum transition. This command only has effect when using the auto_index option to char_library. 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.

Caution: 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 (i.e.: 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

-related <pin>	The related_pin for both forward arcs.
-related_out <pin>	The intermediate transitioning output.
-out <pin>	The last transitioning output.
-cells {list of cells}	List of cells

Use this command to tell 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.

related is the name of the input pin whose transition triggers the arc.

related_out is the first output to switch and becomes the `related_pin` for the out-to-out arc.
out is the output pin for both forward arcs and is the pin in the output library.

This command must be used before **write_library**. Example:

```
define_out_to_out_arc -related A -related_out X -out Y -cells { mycell }
```

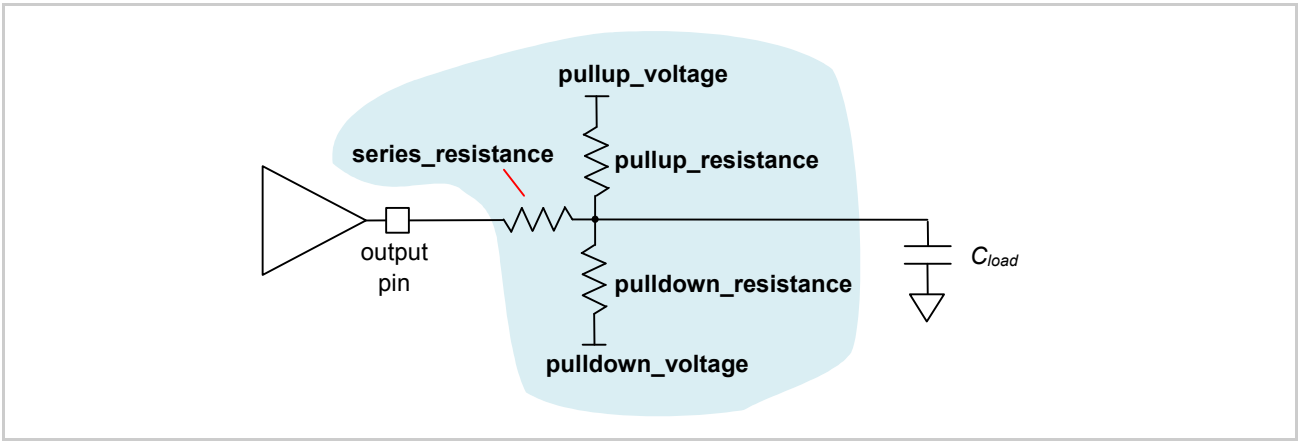
define_pin_load

-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 **define_pin_load** command defines additional loading that can be applied to a particular pin prior to the output load. 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

-when <"function">	Logic conditions of side inputs
-pin {pins}	List of pin names
-related_pin {pins}	List of related pin names
-cell {cell_names}	List of cells

Use this command to identify arcs that generate a pulse on an output pin. This command needs to be executed before running char_library. For the specified arcs Liberate will subtract CV^2 from the fall power. Then Liberate will divide the fall power by two and copy the fall power into the rise power. This is done because the falling arc also captures the rising arc power.

If the `-when` is not specified, then this command is applied to all arcs that match the specified pin and related_pin. If `-when` is specified, then this command is only applied to those arcs

that exactly 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**.

define_template

-type {delay | power | ccs | ccsn_dc | constraint | ecsm | mpw | si_iv_curve | si_immunity }	Template type (REQUIRED)
-index_1 {values}	List values to be used as the first index (REQUIRED)
-index_2 {values}	List values to be used as the second index
-index_3 {values}	List values to be used as the third index
<template>	Name of template

The **define_template** command defines a template to be used for characterization. The **type** argument specifies the type of template being defined. How each cell is to be characterized is defined by associating the defined template with the appropriate argument 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 may 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)
capacitence	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 DC noise model characterization. It requires **index_1** and **index_2** to be specified where they represent a range of input/output voltages. If not specified Liberate uses a range of 29 voltage points from -Vdd to 2*Vdd. DC simulations are very fast, especially on a small CCB group of transistors extracted for noise stage simulations and therefore does 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 29x29. It can be useful to optimize the size of the tables to avoid non-convergence errors at the extremes of the voltage range. The ccsn_dc template is global to all cells and is not included in the define_cell command.

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 **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 the user does not provide an **mpw** template, the **mpw** timing constraint will be characterized as a single attribute. If the variable **mpw_table** is set to a **1**, then Liberate will output a one dimensional mpw table using the **define_cell** constraint index_1 list of values. Use the **define_template type mpw** only when a 2 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.

All **index_*** entries for all the library constructs should be monotonically increasing.

This command must be used before **char_library**.

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
```

get_var

<variable_name> Liberate variable name

This command is used to return the current value of a Liberate variable, either the default value or the value set using **set_var**. A list of Liberate variables can be generated using the **printvars** command. Example:

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

ibis_define_component

-harness_pins {values} A list of harness pins

-manufacturer "value" Component's manufacturer name

-net_typ_vdd {values} A list of {net typ_vdd} pairs

-net_typ_vss {values} A list of {net typ_vss} pairs

-package_C {values} A list of package C at typ/min/max condition

-package_L {values}	A list of package L at typ/min/max condition
-package_R {values}	A list of package R at typ/min/max condition
-sim_fall "value"	Simulation transition fall time in seconds. Default: 1n
-sim_rise "value"	Simulation transition rise time in seconds. Default: 1n
-waveforms {values}	A list of pullup/pulldown waveform templates
<components>	IBIS component name(s) list

The **ibis_define_component** command provides a means for specifying the component definitions that are used for Liberate IBIS characterization.

The **manufacturer** option is used to specify the manufacturer's name in the IBIS output file.

The package_R, package_L and package_C options are used to specify the package R/L/C conditions. These options require a list with 3 values representing the typical, minimum, and maximum values in that order.

The **waveforms** option is used to specify a list of waveform templates. The templates must be defined by the ibis_define_component command.

The **net_typ_vdd** option specifies a list of pairs of nets and typical vdd values. The **net_typ_vss** option specifies a list of pairs of nets and typical vss values. For example, if you are using an IBIS truth table and use the SET_NET_GND command in the truth table to specify the gnd/vss values for the IBIS pin, such as:

```
SET_NET_GND={PAD:0,0,0 PADN:0,0,0}
```

then the following entry is automatically generated:

```
-net_type_vss {PAD 0 PADN 0}
```

Use the **harness_pins** option to specify IBIS pins that have a harness attached. Note: The harness can be used to specify a special pullup/pulldown resistance/voltage setup for differential output pins.

The **component** option specifies a list of IBIS components this command will apply to.

This command must be used before **char_library**.

Example:

```
ibis_define_component \
    -manufacturer "Vendor" \
    -package_R { 0 0 0 } \
    -package_L { 0 0 0 } \
    -package_C { 0 0 0 } \
    -waveforms $wf_templates \
    -net_typ_vdd {A 1.1 PAD 3.3} \
    -net_typ_vss {A 0.0 PAD 0.0} \
{IO_cell_1 IO_cell_2}
```

ibis_define_header

-comment_char <value>	IBIS comment character, default '|'
-copyright <value>	User specified copyright text
-disclaimer <value>	User specified disclaimer text
-file_rev <value>	IBIS file version, Default: '0.1'
-filename <value>	IBIS output file name
-ibis_ver <value>	IBIS version, default '4.2'
-notes <value>	User specified notes
-source <value>	User specified source description
<cellname>	IBIS cell name

The **ibis_define_header** command provides a means for specifying the content of each section of the IBIS header at the top of the IBIS model file.

This command must be used before **char_library**.

Example:

```
ibis_define_header -ibis_ver 4.2 \
    -source "Vendor" \
    -disclaimer "Property of Vendor. IBIS models are provided as a service to
our customers. No liability is assumed for accuracy or proper functioning in
any application." \
    -copyright "Property of Vendor. Unauthorized reproduction and/or
distribution is strictly prohibited. Copyright (C) 2008, Vendor, all rights
reserved." \
    -comment_char "|" \
    -file_rev "1.0" \
    IO_cell_1.ibs
```

ibis_define_model

-cfix	Rising waveform C_fixture
-component	this model is defined for these component(s) list
-cref	Timing specification test load capacitance
-cref_diff	Timing specification differential capacitance
-enable	"Active-High" or "Active-Low"
-lfix	Rising waveform L_fixture
-model_type	Pin model type : Input, Output, I/O, 3-state
-polarity	"Non-Inverting" or "Inverting"
-r_load	Loading resistor for [Ramp] measurement
-rfix	Rising waveform R_fixture
-rref	Timing specification test load resistance
-rref_diff	Timing specification differential resistance

-vfix	Rising waveform V_fixture
-vfix_max	Rising waveform V_fixture_max
-vfix_min	Rising waveform V_fixture_min
-vinh	Minimum upper threshold voltage
-vinl	Maximum lower threshold voltage
-vmeas	Reference voltage for timing measurements
-vref	Timing specification test load voltage
-when	When condition to match waveform's "when" logic
modelName	IBIS model name

The **ibis_define_model** command is required for each IBIS model characterization setup. It is used to define global parameters that are used to acquire and write model data.

This command must be used before **char_library**. Example:

```
set cells {IO_cell_1}
# When 0.8V and 2.00V are measurement thresholds, set Vinl=0.8V and Vinh=2.00V.
ibis_define_model \
    -model_type 3-state \
    -polarity Non-Inverting \
    -enable Active-Low \
    -vinl 0.8 \
    -vinh 2.0 \
    -vmeas 1.65 \
    -vref 0 \
    -rref 100.00M \
    -cref 15p \
    -component $cells \
    IO_cell_1_model
```

ibis_define_model_selector

-component	This model is defined for these component(s) list
-model_desc	List of model and description in pair(s)
name	Model selector name

The **ibis_define_model_selector** command may be used to specify the set of user-defined model types should be included in a specific IBIS model.

This command must be used before **char_library**.

Example:

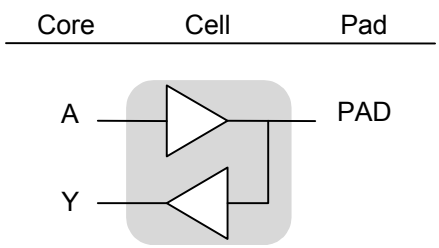
```
set cells {MyOBuff}

set mslist { \
LOW_VOLT_PUPD    "Low Voltage Pull_up enabled"   \
LOW_VOLT_!PUPD   "Low Voltage Pull_down enabled" \
}
```

```
ibis_define_model_selector \
-model_desc $mslist \
-component $cells \
LVDS_OUT
```

ibis_define_pin

-net	Pin net/signal name from Spice netlist
-model	Pin IBIS model name defined in ibis_define_model command
-enable	Enable pin signal (port) name from Spice netlist
-function	'when' condition to set pin to logic 1 (high) state
-inv_pin	The corresponding inverting pin name for I/O output
-inv_pin_net	The inverting pin signal (port) name
-vdiff	The differential receiver threshold voltage from pin to inv_pin
-tdelay_typ	The typ launch delays of the inv_pin relative to the pin
-tdelay_min	The min launch delays of the inv_pin relative to the pin
-tdelay_max	The max launch delays of the inv_pin relative to the pin
-R_pin	Pin package R value
-L_pin	Pin package L value
-C_pin	Pin package C value
-components	IBIS component name(s) list which use this pin definition
-cinpin	Specify the core-input-pin related to this pad pin: core input pin (cell output pin) signal (port) name (eg: pin Y of PAD->Y arc)
-coutpin	Specify the core-output-pin related to this pad pin: core output pin (cell input pin) signal (port) name (eg: pin A of A->PAD arc)
<pin_name>	Pin name

The **ibis_define_pin** command is used to define pins that are required to generate electromigration models.



If PAD is a cell output pin and with arc A->PAD, then A is a core-output-pin (cell input pin). The signal goes from core to IO cell through pin A and then passes to the PAD pin. If PAD is the

cell input pin with arc PAD->Y, then Y is the core-input-pin (cell output pin). The signal goes from PAD to (through IO cell) pin Y and then passes into core.

This command must be used before **char_library**.

Examples:

```
ibis_define_pin \
    -net VDDS \
    -model POWER \
    -component $cells \
    VDDS

ibis_define_pin \
    -net PAD \
    -model IO_cell_1 \
    -enable "GZ" \
    -function "A" \
    -component $cells \
    PAD
```

ibis_define_waveform_template

-type <R | F>	Output waveform direction
-pull_dir <U | D | B>	Resistive load connected direction.
-vfix <value>	Voltage of the fixture
-rfix <value>	Resistance of the fixture
-lfix <value>	Inductance of the fixture
-cfix <value>	Capacitance of the fixture
-pins {list of pins}	List of pins to apply this waveform
-nets {list of nets}	List of net names that correspond to the pins
name	Waveform template name. Suggested names are: pullup_on_data, pullup_off_data, pulldown_on_data, and pulldown_off_data.

The **ibis_define_waveform_template** command is used to define waveforms used by the **ibis_define_component** command.

If **type** and **pull_dir** are not specified, then the waveform template name has to be one of: **pullup_on_data**, **pullup_off_data**, **pulldown_on_data** or **pulldown_off_data**.

The **nets** option is added to provide the net name of the corresponding IBIS pin name.

This command must be used before **char_library**.

Here are details of the fixture parameters, taken from the IBIS 4.2 specification**:**

```
        The 'fixture' subparameters specify the loading conditions
        under which the waveform is taken.  The R_dut, C_dut, and
        L_dut subparameters are analogous to the package parameters
        R_pkg, C_pkg, and L_pkg and are used if the waveform includes
        the effects of pin inductance/capacitance.  The diagram below
        shows the interconnection of these elements.

               PACKAGE             |   TEST FIXTURE
                                   |
 _________
|  DUT    |   L_dut   R_dut       | L_fixture  R_fixture
|  die    |---@@@@@--/\/\/\--o-----|--@@@@---o---/\/\/\----- V_fixture
|_________|                   |     |          |
                              |     |          |
                              |     |          |
                        C_dut ===   |        === C_fixture
                              |     |          |
                              |     |          |
                             GND    |        GND

        NOTE:  The use of L_dut, R_dut, and C_dut is strongly
        discouraged in developing waveform data from simulation
        models.  Some simulators may ignore these parameters because
        they may introduce numerical time constant artifacts.

        Only the R_fixture and V_fixture subparameters are required,
        the rest of the subparameters are optional.  If a subparameter
        is not used, its value defaults to zero.  The subparameters
        must appear in the text after the keyword and before the first
        row of the waveform table.
```

Example:

Here two waveform templates are defined, and both are used by the ibis_define_component

The vfix value should be set differently for individual corners.

```
# In the typ corner (char_typ.tcl), you should have:
set PULLUP_V_PAD 3.3
# In the min corner (char_min.tcl), you should have:
set PULLUP_V_PAD 3.0
# In the max corner (char_max.tcl), you should have:
set PULLUP_V_PAD 3.6
```

The template_ibis.tcl which contains all IBIS related commands (like the following ibis_define_waveform_template) is sourced by all char scripts.

```
# pulldown_on_data
ibis_define_waveform_template \
    -type F \
    -pull_dir U \
    -vfix $PULLUP_V_PAD \
    -rfix 50 \
    -pins "P1" \
    -nets "PAD" \
```

```
        waveform_PAD_F_U_50

    ibis_define_waveform_template \
        -type F \
        -pull_dir D \
        -vfix $PULLDN_V_DOUT \
        -rfix 50 \
        -pins "P3" \
        -nets "DOUT" \
        waveform_DOUT_F_D_50

    ibis_define_waveform_template \
        -type R \
        -pull_dir D \
        -vfix $PULLDN_V_PAD \
        -rfix 50 \
        -pins "P2" \
        -nets "PAD" \
        waveform_PAD_R_D_50

    set wf_templates {
        waveform_DOUT_R_D_50  <not shown above>
        waveform_DOUT_F_U_50  <not shown above>
        waveform_DOUT_R_U_50  <not shown above>
        waveform_DOUT_F_D_50
        waveform_PAD_R_D_50
        waveform_PAD_F_U_50  <not shown above>
        waveform_PAD_R_U_50  <not shown above>
        waveform_PAD_F_D_50  <not shown above>
    }

    ibis_define_component \
        -manufacture "Cadence Design Systems, Inc." \
        -package_R { 0.24 221m 0.35 } \
        -package_L { 2.1nH 4.2nH 3.2nH } \
        -package_C { 2.4pF 2.1pF 2.2pF } \
        -waveforms $wf_templates \
        -sim_rise 1.1 \
        -sim_fall 1.1 \
        -net_typ_vdd { P3 1.32 P2 3.6  } \
        $ibisCellName
```

merge_library

-arcs	Request merging of data by arc.
-cells {cell_names}	List of cell names to merge from the second library. Default: all cells
-driver_waveform	Preserves the driver waveform associations present in the from library
-exclude	Exclude **-cells** from the second library
-filename <filename>	Output file name of merged library. Default: `<to_lib>.m`
-indent <number>	Number of characters to indent the merged library
-libname <lib_name>	Name of merged (resultant) library

-method {method(s) data_type(s)}	Method for merging libraries. Only pertains to certain data types (see below.) Legal methods are min, max, avg, sum, or append (e.g "max" or "delay avg hold max".) Default: "append" (Append data missing from the target library.)
-no_extra_pins	Exclude extra pins from the second library
-types {list}	Merges the specified types. Supported types are: `attributes`, `cap`, `ccs`, `ccs_retain`, `ccsn`, `ccsp`, `constraint`, `delay`, `ecsm`, `ecsmn`, `ecsmp`, `em`, `leakage`, `max_cap`, `max_trans`, `noise`, `min_pulse_width`, `power`, `recovery`, `removal`, `retain`, `minimum_period`. Default: Merge all types.
-unique_pin_data	Output unique timing, power, etc. data for each bus bit or bundle member.
-voltage <voltage>	Create a new library scaled by voltage from two input libraries with different voltages.
<to_lib>	Library to merge to
{<from_lib(s)>}	List of libraries to merge from

merge_library takes two (or more) libraries and merges them to make a new library. It does this by starting with the `to_lib`, and merging data from the `from_lib` into it. (**Note**: the `to_lib` will ***not*** be over-written; 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.

arcs specifies that arc data should also be merged. By default, all library, cell and pin data is merged.

cells and **exclude** arguments can be used to control which cells from the `from_lib` are merged into the `to_lib`. By default all the cells in the `from_lib` will be merged. This option supports the use of a wildcard. If `exclude` is used then the cells in the `cells` list will be excluded and all the other cells included in the merge process.

driver_waveforms preserves the driver waveform associations present in the "from" library. Note that `merge_library` does not preserve any driver waveforms in the "to" library. By default, all references to driver waveforms are dropped from the output library.

indent specifies the number of spaces to use for indentation in the merged library.

libname specifies the name given to the merged library in the `.lib` file.

method specifies the method for merging data into a library. By default merge_library will only append data that is missing in the first library from the second library. With the `method` option, the user can select the `min, max, avg` (average), or `sum` of data for any arcs that are present in both libraries to create the merged library. Arguments are specified as a list of method & type pairs. Legal data types for this method of merging are `delay, retain, power, setup, hold, recovery`, and `removal`. This 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 is used to disable the merging into the `to_lib` any pins that do not already exist in the `to_lib`.

unique_pin_data outputs unique timing, power, etc., data for each bus bit or bundle member. (similar to unique_pin_data for write_library).

Without unique_pin_data

```
bundle (QB) {
    pin (Q0B) {
        ...
    }
    pin (Q1B) {
        ...
    }
    timing () {
        ...
    }
}
```

With unique_pin_data

```
bundle (QB) {
    pin (Q0B) {
        ...
        timing () {
            ...
        }
    }
    pin (Q1B) {
        ...
        timing () {
            ...
        }
    }
}
```

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 will be 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 will be inserted in the output library to indicate that merging with voltage scaling was performed.

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.

the "target" library; the data from this library will form the initial data in the new library, and data from the `from_lib` will be merged into it.

from_lib(s) specifies one or more libraries that data will be taken from and merged into the "target" (to_lib). If more than one library is specified, the merging will take 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.

Important: Merging libraries should be performed as a separate Liberate run.

Example:

```
# Merge CCS into an existing library
merge_library -filename orig_ccs.lib orig.lib ccs.lib
```

one_cold

{ list } Specify a list of one-cold pins (Default: none)

This Tcl function takes a list of pins and will return 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)).

This command must be used before **char_library**.

Example:

```
define_cell -when [ one_cold { a b c } ] …
```

one_hot

{ list } Specify a list of one-hot pins (Default: none)

This Tcl function takes a list of pins and will return 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)).

This command must be used before **char_library**. Example:

```
define_cell -when [ one_hot { a b c } ] …
```

printvars

The **printvars** command lists the current values of all of Liberate's command variables. This command does not have any options.

Example:

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

packet_slave_cells

This is a Tcl function that returns a unique list of cells assigned to a particular slave.

When called from the server, an empty list will be returned. See Parallel Processing

read_ldb

-check_driver_waveforms	Checks both pre-driver and active-driver generated waveforms in ldb to verify that they have the correct slews.
-check_spice	Check netlist and model for changes and recharacterize cell if any changes are detected.
-check_var	Checks variables and commands for changes. If detected, the appropriate data type will be recharacterized.
-incremental	Specifies incremental capability.
-remove {list of cells}	List of cells to remove from the ldb. Default: (none)
-remove_failed	Remove cells that are marked as failed in the ldb.
-remove_type {list of types}	List of data types to remove from ldb. Default: (none)
<filename>	A library database file in ldb format.

The **read_ldb** command reads an existing library database (ldb) created by the `write_ldb` command. The library database can then be used for formatting the library data, for example, creating a datasheet.

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`.

You may have a given variable stored in the ldb and also specified in your Tcl script. Whichever occurs *last* in your script will determine the value of the variable (i.e.: the last setting of the variable 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.

incremental instructs Liberate to examine the ldb and perform characterization for any data that is missing from the ldb. (See below for `check_var` and `check_spice`**.**)

remove specifies a list of cells to remove from the ldb. 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 re-characterized.

remove_failed will 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 specifies a list of data types to be removed from the ldb, so they will be re-characterized. Supported types are:

ccs ccsn ccsp cin constraint delay hold leakage mpw power setup

check_var checks changes in control variables and commands. If a change is detected, that change will map to a specific `remove_type`, and that type (or types) will be recharacterized. This only works if **-incremental** is set.

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 variable enable_command_history
- set variable 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 re-characterized.
- Re-simulating only constraint arcs (for example, if constraint_delay_degrade was changed). This will be faster than re-characterizing 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.

Example:

```
# Read an ldb to generate additional formats
read_ldb tt_all.ldb
# Write a HTML datasheet
write_datasheet -format html -dir tt_html tt
# Write .lib
write_library tt_all.lib
```

Note: When an ldb is read in, if the `scale_tran_by_template` and/or `scale_load_by_template` variables were set to 1 when the VDB was created, they will be reset to 0 to prevent double scaling of the indices.

read_library

{library_names} List of library file(s) to read in Liberty format.

The **read_library** command reads an existing Liberty format library into memory. 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

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
```

read_spice

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

{<spice_netlist_file>} List of file(s) with extracted circuit netlists in SPICE format

The **read_spice** command reads in the SPICE netlists of the cells. The model files should be included with the circuit netlists. The SPICE netlist and model formats supported by Liberate are as follows:

- Hspice netlist, Level=49, Level=53 and Level=54 models,

- Spice3 netlist, BSIM3 and BSIM4 models. Well proximity effects are supported.
- PSP
- eldo, used to load eldo format models

This command must be used before **char_library**.

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_truth_table

{<filename>}	List of Truth Table files which contain one or more truth tables to load.

This command reads in and validates truth-table files. This command must be used before calling **write_template**. Use the **write_template** command with the **truth_table** and the **auto_index** options to output a template file built from the loaded truth table file data.

The **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** arguments 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

{<filename>} The name of a VDB file to load.

This command 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. 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.

remove_pin_attribute

{cells} List of cells

{pins} List of pins

{attribute list} List of attributes

Use this command to remove automatically generated pin-level attributes from the output Liberty file. Pin level attributes can be provided in the **write_library -user_data** file or automatically added by Liberate. Note: To remove pin level attributes that are supplied in the user_data file, modify the user_data file.

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}
```

reset_defaults

-version <version> Default: Reset all variables needed to restore the behavior in the previous release.

From release to release, defaults values of some Liberate variables might change. This command restores the changed variable settings to 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. This command should be used at the start of your Tcl file and can only be used once per run.

Examples:

```
Example:
# Reset Liberate defaults to 12.1
reset_defaults -version 12.1
```

set_aging_criteria

-attribute <value>	Settings about reliability analyses. (REQUIRED)
-duty_cycle <value>	The stress simulation duty cycle that 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%).
-period <value>	Stress simulation period. (REQUIRED)

This command 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. You may get incorrect aging characterization data if you use an earlier version of the MMSIM release. In addition, use the **extsim_model_include** and **define_leafcell** command with **set_aging_criteria**.

The **-attribute** argument 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 argument is required.

The **-duty_cycle** argument specifies the duty cycle that is applied to the period for stress simulation. If this argument is not specified, the default value of 0.50 (50%) is applied to netlist aging.

The **-period** argument specifies the period of stress simulation (in MKS units). There is no default period. This argument is required.

This command must be used before **char_library**.

Example:

```
# Set the aging criteria
set_aging_criteria \
     -period 300e-9 \
     -duty_cycle 0.5 \
     -attribute "   age time = \[10y\]\n   deltad value = 0.1\n
report_model_param value=yes\n"
```

set_client

-dir <directory_name>{%N%U%P%S}
Directory for temporary files (REQUIRED)

<machine_name> Name of client machine or queue name

The set_client command specifies a client a machine to be used for distributed library characterization.

The **dir** argument 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.

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** argument supports a deprecated method of using a queuing system such as LSF. For more details on this option, see Appendix D, “Deprecated and Legacy Variables.”

set_conditional

-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 to disable conditional arcs for delay and constraint groups for a list of cells. Currently other arc types such as power are not supported. This command can be used after **char_library** but before model generation.

Examples:

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

set_constraint

-cells {cell_names}	List of cell names. Default: none
-index1_factor <value>	Multiplication factor. Default: 0.0
-index2_factor <value>	Multiplication factor. Default: 0.0
-margin <value>	Margin to add to timing constraints. Default: 0
-max <value>	Maximum constraint value. Default: 1e20
-max_margin <value>	Maximum margin allowed for the given constraint. Default: 1e20
-max_risefall	Specify to use the max from the rise and fall constraint
-min <value>	Minimum constraint value. Default: -1e20
-min_margin <value>	Minimum margin allowed for the given constraint. Default: -1e20
-min_recrem <value>	Minimum sum of *recovery + removal.* Default: *no checking*
-min_setuphold <value>	Minimum sum of *setup + hold.* Default: *no checking*
-min_warning <0 | 1 | 2>	Warning level when ***min_recrem*** or ***min_setuphold*** are exceeded. Default: 1
-non_seq_sum_swap_direction	Sum non_seq_setup/non_seq_hold using the opposite direction to that used for summing setup/hold. Default: follow "sum_same_direction".
-pin_dir <rise | fall | both>	Specify which constraints to apply the margin. Default: both
-pin_probe_factor	Apply when constraints are measured using path delay. Default: 0
-pins {pin_names}	List of pin names. Default: none
-related {list}	List of related pins.
-rel_margin <value>	Margin ratio to add to timing constraints. Default: 0
-related_probe_factor	Apply when constraints are measured using path delay. Default: 0
-sum_same_direction	Request summing using the same direction. Default: opposite direction.
-type {setup | hold | recovery | removal | mpw}	Timing constraint type.
-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, 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.

The **min_margin** and **max_margin** arguments specify the minimum and maximum margin allowed for the given constraint type. The **pin_dir** argument specifies whether the margin is to be applied to just **rise** constraints, **fall** constraints, or **both** rising and falling. The **related_probe_factor** and **pin_probe_factor** are factors that are applied to the final margin, based on the measurement of the pin to probe delay and the related pin to probe delay. These only apply when constraints are measured using path delay. 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).

The **min** and **max** options can be used to specify a minimum and maximum constraint value. The default minimum constraint value is -1e20. The default maximum constraint value is 1e20.

The **min_recrem** and **min_setuphold** options specify a minimum value (in seconds) that the respective sums must exceed. If the minimum is not met, the removal and/or hold values are adjusted as required to meet the minimum. When the **min_recrem** and/or **min_setuphold** are not met, the warning level can be controlled by setting **min_warning** as follow: **0** = no warnings; **1** = 1 warning per table; **2** = 1 warning per table value.

The **sum_same_direction** option can be used to check min_setuphold and min_recrem. Specify this option to use the same transition for checking the minimum sum of setup/recovery with hold/removal. The default is false (sum rise + fall, fall + rise).

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.

The **max_risefall** option requests the maximum value between the rise_constraint and fall_constraint.

When the **type** of **setup** and/or **hold** options enable 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
```

The **pins** and **cells** options specify which pins/cells the set_constraint command will be applied to. The pins and cells options currently only apply to the **index1_factor** and **index2_factor** options.

To have an effect on the output library, this command must be executed before **write_library**. If the **define_arc pin_probe** and **related_probe** options are used and a margin is desired for the delay measurements that will be reported in the ldb, then 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

-cells {cell_names} List of cell names. Default: *all cells*

-delay_degrade <value> Delay degradation relative tolerance ratio. Default: 0.1 (10%)

-delay_degrade_abstol <value>
Delay degradation (minimum) absolute tolerance in seconds. Default: 5e-12 (5 ps)

-delay_degrade_abstol_max <value>
Delay degradation maximum absolute tolerance, in seconds. Default: -1 (off)

-glitch_peak <value> Glitch peak relative tolerance ratio. Default: `0.1`

-metric <list> Specifies the constraint metric type. The supported values are: `delay`, `glitch`, `slew`, `width`, or `path_delta`.

-min_constraint {cell pin ...}
Specifies that the minimum constraint (instead of the maximum constraint) will be written to the library.

-output_load <min | max | value | index_?>
Specifies the output load applied during constraint characterization. The value must be in Farads. Default: min (from `constraint_output_load`).

-pin {pins} List of pins. Default {} (no pins)

-pin_dir <R | F> The switching direction of the pin.

-probe {probes} List of probe nodes

-related_pin {pins} List of related_pins

-related_pin_dir <R | F> The switching direction of the related pin.

-slew_degrade <value> Slew degradation relative tolerance ratio. Default: -1 (not enabled))

-type < setup | hold | recovery | removal | mpw | min_pulse_width | non_seq_setup | non_seq_hold >
Type of constraint to which the criteria applies

-width_degrade <value> Specifies pulse-width degradation ratio. Default: follows the `constraint_width_degrade` variable.

The **set_constraint_criteria** command 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. To set the global constraint related criteria do not include any of the arc-specific settings such as the arguments `-cells`, `-type`, `-pin`, `-pin_dir`, `-related_pin`, and `-related_pin_dir`. These arguments (`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 then 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 variables, but if both the global variable 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, then the order of precedence is:

- 1st: define_arc
- 2nd: set_constraint_criteria
- 3rd: global variable

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, then the specified value is used in place of the global parameter value.

cells specifies a list of cells that the specified criteria will affect. If the `cells` list is empty or not included, then the specified criteria are applied globally.

delay_degrade specifies the maximum amount of delay degradation 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. The delay_degrade is percentage, specified as a positive decimal (usually from 0.0 to 1.0). This option will override the global variable constraint_delay_degrade when the `cells` option is not used.

delay_degrade_abstol specifies the minimum delay degradation value permitted (in seconds) in the clock-to-constraint output pin (flip-flop) or the data-to-constraint_output_pin (latch) delay. This option will override the global variable constraint_delay_degrade_abstol when the cells option is not used. A constraint bisection search bound will be 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 specifies the maximum absolute tolerance (in Seconds). The delay_degrade_abstol and delay_degrade_abstol_max when specified together set both an upper and a lower bound to the delay degradation. This option will override the global variable constraint_delay_degrade_abstol_max when the `cells` option is not used.

glitch_peak specifies the maximum size of logic glitch permitted on the probe node before an arriving signal is deemed to fail a timing constraint. This option will override the global variable constraint_glitch_peak when the `cells` option is not used.

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

min_constraint instructs Liberate to put in the library the minimum constraint (instead of the maximum constraint) for the specified cell-pin pairs. This command argument 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 specifies the load to be applied to cell output pins during constraint characterization. This can be used, for example for removing glitch suppression for cells with an expected output glitch from the clock when a reset or set pin is active. This option will override the global variable `constraint_output_load` when the cells option is not used. See the variable constraint_output_load for supported settings.

pin specifies a list of 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 specifies the switching direction of the pin.

probe specifies a list of probe nodes to be used for measuring the constraint. A constraint is characterized by measuring a delay from the related_pin to the probe or by measuring a glitch on the probe. If this argument is specified, then the `-probe` option of the **define_arc** command is not needed. The `-probe` option of the **define_arc** command takes precedence over the `-probe` option of the **set_constraint_criteria** command.

related_pin specifies a list of related_pin(s) for the constraint. The related pin is typically a clock pin.

related_pin_dir specifies the switching direction of the related_pin.

slew_degrade includes slew degradation when determining timing constraints. By setting the slew criteria both the delay degradation and the slew degradation will be checked and the first criteria to fail will determine 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_*** variables. This option overrides the global variable constraint_slew_degrade when the cells option is not used.

type specifies the constraint type to which the criteria will be applied. The supported types are:

```
setup
hold
recovery
removal
mpw
min_pulse_width
non_seq_setup
non_seq_hold
```

This option will be ignored if the `cells` option is not specified.

width_degrade 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 is used together with the `-metric width` option of the define_arc command. This option will override the global variable constraint_width_degrade when the cells option is not used.

Example:

```
# Set the hold criteria to 10% glitch for clock_gater
set_constraint_criteria \
    -type hold \
    -glitch_peak 0.1 \
    -cells clock_gater

# Set the setup criteria to 10% delay degradation and
# 50% slew degradation for the clock_gater
set_constraint_criteria \
    -type setup \
    -delay_degrade 0.1 \
    -slew_degrade 0.5 \
```

```
    -cells clock_gater

# set the global criteria
set_constraint_criteria  \
    -delay_degrade 0.15 \
    -delay_degrade_abstol 10e-12 \
    -glitch_peak 0.5 \
    -slew_degrade 0.5
```

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 arguments 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.

Example:

```
# 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 }
```

This command must be used before **char_library**.

set_default_group

-cells {cell_names} List of cell names (Default: *all cells*)

-criteria {<delay | power | leakage | cap> <off | force_off | min | avg | max>}
List of matched pairs (Default: {} – use max for all criteria except leakage which uses avg)

-method {<default | const> <bitwise | table>}
List of matched pairs (Default: *{} – use* bitwise *for cells*)

-unateness <merge | separate>
Desired unateness (Default: *merge*)

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

method 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 variable and, if used globally, will overwrite the value of that variable. There are 2 algorithms available when requesting an avg default power group. For more information, see the default_power_avg_mode variable.

unateness keeps positive_unate and negative_unate timing groups from being merged in the default group. By default, if both positive_unate and negative_unate timing groups exist, they will be merged into a single non_unate default group. This option accepts values of merge or separate. The default unateness value is merge. This option only applies to default timing groups and will have no effect on other data groups. (Note: If all timing_sense attributes are identical, then the original timing sense will remain unchanged during merging. This option replaces the default_unateness variable and, if used globally, will overwrite the value of that variable.)

The **criteria** option can be used to specify 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. This option replaces the **default_timing** variable and, if used globally, will overwrite the value of that variable. The available types, values and their default values are:

Type	Values	Default
delay	off | force_off | min | max	max
power	off | min | avg | max	max
leakage	min | avg | max	avg
cap	min | avg | max	max

The value of **force_off** will tell Liberate to remove one of the rise/fall delay groups which already have state dependent groups from the default timing group. This value is supported in the global mode only. That is, the **force_off** option cannot be used together with the **cells** option.

The **cells** option can be used to specify a list of the cells that this command will apply to. If omitted, this command will apply to all cells globally.

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 will override any previous settings for that cell.

Important Notes:

- 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 variables will not be supported. We strongly recommend this command be utilized *instead* of the global variables.
- This functionality does *not* support the ability to read in a library database (ldb), modify the default group settings and write out a modified library.
- The recommended settings for Liberate are to use default groups for NLDM models, but *not* CCS or ECSM models.

This command must be used before **char_library**. Example:

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

set_dependent_load

-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 is the maximum delay load index value
load_value is the 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 will specify 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.

This command must be used before **char_library**. Example:

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

set_driver_cell

-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. Ignores -input_transition option

-instantiate Instantiate the driver cell in the simulation decks

-pin_map {<driver_pin>, <cell_pin>} List of driver_cell_pin to char_cell_pin pairs

-pinlist {<cell> <pin>} List of cell/pin pairs

<driver_cell> Name of driver cell

By default, Liberate uses a linear ramp as the input waveform during characterization. The `set_driver_cell` command defines an active pre-driver cell to be used instead of the linear waveform. This pre-driver cell is driven at its input by a linear ramp defined by the **input_transition** argument. 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).

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

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.

accuracy_mode enables an algorithm for generating waveforms while observing the slew behavior. The default and recommended value is 1.

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 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 variable `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 variable 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, then the **pin_map** and pinlist options are also required.

pinlist specifies a list of pin pairs between driver cell output pins and characterization cell input pins. The `pin_map` option is a list that specifies to which cell the pin pairs in the `pinlist` are mapped. The `pin_map` will map by position to the `pinlist` pins with the first `pin_map` cell corresponding to the first `pinlist` pair, etc.

If a `pinlist` is given then the driver cell is used for only the specific cell and pin pairs in the `pinlist`. The cell names can be wild-carded with a *.

Currently there is no limit to the number of `set_driver_cell` commands specified.

This command must be used before **char_library**. 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

-slew_select <`all` **|** `hybrid` **|** `index`**>**
Specify which slews to store. Default: `all`

<file name> Specify the name of the file where driver waveforms are stored by the server in packet flow. (Required)

This command is used to specify a filename that will hold 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. This flow cannot be used with auto-index because the client determines the slews for the cell, and the server does not know all the indices before the client characterizes the cell.

Use `slew_select` to specify which waveforms to store for reuse by the clients. 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 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.

This command must be used before **char_library**.

set_gnd / set_vdd

-attributes {name value ... } List of attributes specified as name-value pairs to include in the pg_pin group. Default: none

-cells Specify a list of cells.

-combine_rail Request summing of power from include list into this supply net.

-ignore_power Ignore the power contribution from this supply net**.**

-include { list }	List of power rails to merge power from. Default: {}
-name_map <value>	Map the supply pin to a different named supply.
-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)

Note: A warning is generated if **set_gnd** sets a pin to large positive voltage or **set_vdd** sets a pin to 0 volts.

The **set_gnd** and **set_vdd** commands identify the **net_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. It is strongly recommended that all supply pins are identified using **set_gnd** and **set_vdd**, as appropriate.

attributes provides for adding a list of attributes specified as name-value pairs. 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 variable.

If **ignore_power** is set, the contribution of the specified supply net will be ignored. That is, the current in this supply net will not be summed into any power measurement. The **ignore_power** option will be skipped if the **cells** option is specified.

type specifies the supply type: primary, backup, internal, nwell, pwell, deepnwell, deeppwell

no_model tells Liberate not to include this supply in the output **.lib** file.

cells provides a list of cells that 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 ("_").

The **-name_map** option specifies the name that this supply will be called in the `output .lib` file. Name mapping is only supported when the **pg_pin** (see: **pin_based_power**) 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 vdd1
set_vdd -cells {INV_X1}   -name_map VDD_X1   VDD 0.9
set_vdd -cells {INV_X16}  -name_map VDD_X16  VDD 0.8
```

combine_rail combines the power and leakage from the supplies specified in the **include** option to 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_vdd/set_gnd** commands.

include provides a list of power rails to be merged into the power rail specified in the **set_vdd/gnd** command. To merge the power, the **combine_rail** option must also be specified.

Liberate will automatically identify: **0**, **GND**, and **VSS** (case insensitive) as ground supplies and will set 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** since this is considered the reference ground.

Liberate will automatically identify the **VDD** (case insensitive) power supplies and will 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 will generate a warning if set_vdd is used to set a pin to 0 volts.

Example:

```
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 (i.e. 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
```

This command should be used before the **read_spice** command to have the desired effect.

This command must be used before **char_library**.

set_input_voltage

-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

Creates `input_voltage` groups at the library and pin level. For a default group the naming convention is: (default_$vdd_$gnd_$direction). Example:

```
input_voltage(default_VDD1_VSS1_output)
```

For multiple groups, the naming convention is: (user_voltage_$num).
Example:

```
input_voltage(user_voltage_0)
input_voltage(user_voltage_1)
input_voltage(user_voltage_2)
```

set_logic_condition

-cells {list}	List of cell names
-pin {list}	List of pin names
-related_pin {list}	List of related pin names
-type {list}	Supported types are constraint, delay, hidden, leakage, mpw
logic_condition {string}	Logic to be applied to filter vectors

Allows the insertion of logic conditions into arcs to reduce/filter vectors that are to be ignored. This command will add the -logic_condition (see define_arc -logic_condition) option to any define_arc commands that overlap with the option settings of this command that are not to reduce the vector space, and control the number of simulations without modifying a verbose template.

Example:

```
set_logic_condition \
  -type delay \
  -pin "Z" \
  -related_pin "A1" \
  -cells { ao32 }
  "B1 & C"
```

This command must be used before **char_library**.

set_max_fanout

-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

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.
```

set_network_port

<port_number>	Network port number

The **set_network_port** command defines an explicit network port number to be used for distributed library creation. 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 on distributed parallel processing see Chapter 6.

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

-name <value>	Specify the operating condition name
-temp <value>	Temperature to characterize at in °Celsius (REQUIRED)
-voltage <value>	Default power supply voltage in volts (REQUIRED)

The **set_operating_condition** command 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.

The **name** option specifies the operating condition name to be used in the output library.

The **temp** argument defines the temperature while the **voltage** argument specifies the default positive supply voltage. This voltage will be assigned to any **VDD** pin name. The default negative supply voltage is 0V. To specify additional power or ground supply nets and their appropriate values use the **set_gnd / set_vdd** 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 **char_library**.

Example:

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

set_output_voltage

-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

Creates `ouput_voltage` groups at the library and pin level. For a default group the naming convention is: (default_$vdd_$gnd_$direction). Example:

```
output_voltage(default_VDD1_VSS1_output)
```

For multiple groups, the naming convention is: (user_voltage_$num).
Example:

```
output_voltage(user_voltage_0)
output_voltage(user_voltage_1)
output_voltage(user_voltage_2)
```

set_pin_attribute

-abstol <value>	Absolute threshold in seconds. Example: 1e-12. (Required)
-attributes {list}	List of attributes. Default "".
-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. Example: 0.05 for 5%. (Required)
-type "string"	The type of data to be modified. Currently, it accepts only `"min_max_cap_tran"`. (Required)
-when "string"	The arc conditional state, that is, the *when* condition.

This command modifies pin attributes in the library.

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.

abstol is the absolute threshold that is used to compare the result of:

```
abs(cell_rise/cell_fall)
```

The specified absolute threshold value must not be negative. For example, use 1e-12 for 1ps.

attribute specifies the attributes that will be updated. When the **-type** is set to `min_max_cap_tran`, the supported attributes are: `max_transition`, `min_transition`, `max_capacitance`, and `min_capacitance`.

The argument can be set to these or if not set, will default to these attributes.

reltol is the relative threshold that is used to compare the result of:

```
abs(cell_rise - cell_fall) / (cell_rise+cell_fall)
```

The specified relative threshold value must not be negative, but must be less than 1. For example, use 0.05 for 5%.

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.

For 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

-side_input <controlling | noncontrolling | all>
How to select different groups of vectors (Default: all)

-range_use_side_input When set, the capacitance range will observe the side_input setting.

-state <min | avg | max> How to select between different states (Default: *max*)

-state_avg_method <merge | separate >
Select the method used to compute the avg capacitance (Default: *merge*)

-table <min | avg | max>	How to select within a table (Default: *max*)
-direction <min | avg | max>	How to select between rise and fall capacitance (Default: *max*)
-meas_supply_cap	Measure capacitance on supply pin.
-when <"function">	Logic conditions of side inputs
-vector <"vector">	Logic conditions of side inputs must match define_cell -pinlist.
-pin {pins}	List of pin names
-pin_dir < rise | fall | both >	Specify the pin direction
-cell <cell>	cellname

The **set_pin_capacitance** command specifies how the simple *capacitance, rise_capacitance* and *fall_capacitance* attributes for each pin are determined. By default, the maximum value of all the calculated pin capacitances is used regardless of direction (rise or fall), logic state or slew/load in the characterized state-dependent capacitance tables. Using this command enables numerous selection levels. The first level (**state**) determines whether to use the minimum (**min**), average (**avg**) or the default maximum (**max**) values from all the tables amongst each logic state (*when* condition). The next level defines which entry in the consolidated table from the previous step should be used (**min**, **avg** or **max** (default)). From these selections a single value for *rise_capacitance* and a single value for *fall_capacitance* will be extracted. The final level of refinement determines how the *capacitance* attribute is determined from these two values. Again, the **min/avg/max** values can be selected, default is **max.** This command can be used independently from characterization, only impacting the attributes output by **write_library** as the characterization database (ldb) will contain all the rise/fall state-dependent capacitance tables for each pin.

Use the option **state_avg_method** to select the algorithm used to compute the average capacitance. For the **state_avg_method** to have any effect, the **set_pin_capacitance -state** option must have a value of **avg** and there must be timing groups in the .lib 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
```

Use the option **range_use_side_input** to change the *capacitance_range* calculation to observe the **side_input** 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
```

This option **side_input** accepts a setting of **controlling**, **noncontrolling**, and **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.

meas_supply_cap causes Liberate to take a capacitance measurement on the supply pin. The measurement is made using the leakage deck. (Note: it does not affect leakage characterization.) The power supply cap 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 cap will be written to the output library as a commented pin group containing a capacitance attribute.

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
```

-vector should be consistent with the `-pin_dir` for the ports listed in the `-pins` option. If the **-vector** has an "x" for the `-pins`, then Liberate enforces consistency. If the **-vector** has an R or F, then the `-pin_dir` must have rise or fall. There is no character supported in the **-vector** to indicate both directions. That is, there is no support for a "b" in the **-vector**.

set_pin_delay_threshold

-cells { list }	List of cells (REQUIRED)
-pins { list }	List of pins (REQUIRED)
{ values }	Specify two delay measurement thresholds: input_threshold_percent_fall, input_threshold_percent_fall

This 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). Example:

```
set_pin_delay_threshold -cells {cellA cellB} -pins {pin1 pin2} {.3 .2}
```

set_pin_gnd

-add_supply	Create and add **supply_name** to cell gnd list.
-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)

(See set_pin_vdd for description of arguments.)

set_pin_slew_threshold

-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, upper_fall (REQUIRED).

The **set_pin_slew_threshold** command 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** parameters and **define_arc –slew_thresh** for the specific cell(s) and pin(s).

set_pin_vdd

-add_supply	Create and add **supply_name** to cell vdd list.
-leakage_add_to_supply <name>	Add the leakage for this port to the named supply.
-supply_name <name>	Name of the supply that drives this pin. (REQUIRED)
{cell_names}	List of cells (REQUIRED)
{pin_names}	List of pin names. For example, {a1 a2 a3 c din ckn}, or regexp like {a* c*}. (REQUIRED)
<vdd_value>	Power supply value (REQUIRED)

The **set_pin_vdd** and **set_pin_gnd** commands associate a **pin** of a cell with a particular supply voltage. These commands are useful for setting power supplies on cells that have multiple power connections, such as level shifters. The cell and pin options will accept 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.

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

When the **supply_name** option is used, and **voltage_map** is set to **1**, then the specified **supply_name** will be output in *related_power_pin/related_ground_pin* format. If **voltage_map** is set to **2**, then the *input/output_signal_level* attributes will be used instead.

leakage_add_to_supply 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.

add_supply is a flag that instructs Liberate to create a new supply with the name specified by the supply_name option (if the supply doesn't already exist.)

Cautions

1. If there is 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 a 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

# Voltage source 3.3 doesn't exist:
set_pin_vdd busDriver PAD 3.3
```

This command must be used before **char_library**.

Example 1:

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

Example 2:

```
set_vdd VDD $VOLT1
set_vdd VDD1 $VOLT
set_pin_vdd -supply_name VDD1 -leakage_add_to_supply VDD \
    LVLHLD1HVTIN$VOLT
```

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

set_receiver_cap_thresholds

-rise {list}	List of receiver rise capacitance thresholds specified as a percentage in decimal. (i.e: .5 = 50%)
-fall {list}	List of receiver fall capacitance thresholds specified as a percentage in decimal. (i.e: .5 = 50%)

This command allows users to override the default behavior of Liberate with respect to ecsm_capacitance tables.

For **1-piece tables**, Liberate will report 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 users to augment these tables with additional voltage thresholds.

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 ecsm_capacitance tables for 20%, 30%, %50, 70%, 80%
# for both rising and 30%, 50%, 70% for falling arcs
#
set_receiver_cap_thresholds \
    -rise {0.2 0.3 0.5 0.7 0.8} \
    -fall {0.3 0.5 0.7}
```

set_rsh_cmd

{rsh_cmd_string}	String to be used for the rsh_cmd
{cells}	List of cells to which this rsh_cmd is applied

This command 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.)

Notes:

- 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 variable 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

-cells {list}	List of cell names
-floating_node < init | ignore >	Initialization for floating nodes. Default: init
-method < hybrid | ic | ic_except_dc >	Method for setting initial conditions. Default: ic
-vector_skip_open_source_drain	Skip floating low(high) state if no pull down(up) path

Use this command to tell Liberate how to set initial conditions when presenting spice decks to external simulators, (i.e., it allows local/global control of **sim_init_condition**.) See also **char_library -extsim**.

cells is a list of cells to apply this command to. Default: all cells

floating_node is a flag that tells Liberate to initialize (default) floating nodes, or to ignore floating nodes. When ignored, the external simulator can initialize the nodes or the user can provide the initialization using the define_arc command option -extsim_deck_header. 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 specifies the method for initializing nodes. (This is the same as the variable sim_init_condition.) Accepted values are:

hybrid	Use `.ic` for any floating nodes and `.nodeset` everywhere else.
ic	Always use `.ic` to initialize nodes in the external simulation. (Default)
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.

vector_skip_open_source_drain is a flag to tell 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. 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 flag vector_skip_open_source_drain is used, the vector with N1=1 and N2=0 will be skipped.

This command must be used before **char_library**.

set_simultaneous_switch

-off	Disable simultaneous_switch on the list of cells.
{cell_names}	Names of cells (REQUIRED).

This command can be used to disable simultaneous switching input analysis on a cell-by-cell basis. See the simultaneous_switch variable for more information about simultaneous switching input analysis. Example:

```
set_simultaneous_switch -off { mycell }
```

set_three_state

-current_degradation_threshold <value>	Specify the degradation ratio,. (Default: 0.1)
-off	Turn off three state modeling

{ }	List of cells

Use this command to disable 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.

Use the **current_degradation_threshold** option to specify a disable arc current degradation threshold. 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).

This command can be used after **char_library** except in one case. If the **current_degradation_threshold** is specified, then this command must be used before **char_library**. Example:

```
set_three_state -off { my list of cells }
```

set_units

-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 is 1kohm.
-timing < 1ns | 100ps | 10ps | 1ps >	Specify the timing units. (Default: 1ns)

This command 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 may 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.

This variable must be set before any command that creates a library, such as **write_library**, or **write_verilog**.

may be set after **char_library**. Example:

```
# Set the timing units to 1 pico second.
set_units -timing 1ps
```

set_var

<parameter>	Parameter name (REQUIRED)
<value>	Parameter value (REQUIRED)

The **set_var** command is used to set Liberate-specific parameters. The available Liberate parameters are defined in Chapter 4. Example:

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

set_vdd

See set_gnd / set_vdd for details on this command.

write_datasheet

-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"
-groups	Writes only specified cell groups to the datasheet
-include_indices	Include indices 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-last. Default: first-mid-last
<library_name>	Library name for the datasheet and filename prefix

The **write_datasheet** command writes a datasheet in the format specified by the `format` argument. If the format is `text` the datasheet is written to a file named `filename` unless no filename argument is specified in which case the datasheet is written to a file called `library_name.txt`. The `write_datasheet` command can be called after a `char_library`, `read_ldb` or `read_library` command.

Sample output

```
Delay(ns) to Q rising (conditional):
Cell Name    Timing Arc(Dir)    When       Delay(ns):  Min       Mid       Max
DFF3T        CLK->Q (RR)        -                      0.0261    0.0465    0.1452
             SET->Q (FR)        CLK&D                  0.0439    0.0660    0.2019
             SET->Q (FR)        CLK&!D                 0.0458    0.0659    0.2026
```

```
SET->Q (FR)        !CLK&D                  0.0324     0.0533     0.1671
```

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 then 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** argument is specified then 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>.gif* then it will also be included. The datasheet writer is written using Liberate's Tcl API and is available for customization in the file `$ALTOSHOME/etc/datasheet.tcl`.

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 web. The recommended setting is html. The **write_datasheet** command must be used after **char_library** and **read_ldb**.

The option **include_indices** can be specified to request Liberate to include the indices 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.

map option provides for mapping internal signal names to an external name. Arguments 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). 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_ibis_file

-append_file	Switch that says to append to existing file instead of writing a new file. Default: off (don't append)
-gndclamp_max <value>	Power Clamp max cutoff. Default: 1e-6
-gndclamp_min <value>	Power Clamp min cutoff. Default: -1e-6
-no_header	Switch that says to output only the [Model] section. Default: off (print all sections.)
-pulldn_max <value>	Pull-up max cutoff. Default: 1e-6
-pulldn_min <value>	Pull-up min cutoff. Default: -1e-6
-pullup_max <value>	Pull-up max cutoff. Default: 1e-6
-pullup_min <value>	Pull-up min cutoff. Default: -1e-6
-pwrclamp_max <value>	Power Clamp max cutoff. Default: 1e-6
-pwrclamp_min <value>	Power Clamp min cutoff. Default: -1e-6
-rev <value>	IBIS file revision. Default: "v1.0"
-versions { }	IBIS versions. Default: "4.2"
files { filenames }	List of IBIS files to create
ldbs {ldb filenames}	List of ldb files for typ/min/max corners
cells {cell names}	List of cell names

The **write_ibis_file** command writes an IBIS formatted model. See the section on IBIS file creation in this manual for more information.

append_file tells Liberate to append to an existing file, rather than create a new file. Must be used with no_header.

no_header tells Liberate to create just the [Model] section, and to omit the file header, the [Component] section, and the [Model Selector] section. This option together with append_file are used to create an IBIS file with multiple [Model] sections for a single device.
(See Creating an IBIS file with multiple [Model] sections for a single device.)

The **pwrclamp***, **gndclamp***, **pullup*** and **pulldn*** options can be used to filter data from the output .ibs file that may not be desired.

write_ldb

<filename> A library database file in **ldb** format

The **write_ldb** command 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.

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.

It is recommended that the **write_ldb** command be executed immediately following the **char_library** command and before any model creation commands such as **write_library**. This is highly recommended so there is a clean, unmodified copy of the ldb saved for future use. This is important because, for example, when user data is loaded with **write_library user_data**, the internal database will be modified by the user data and any ldb subsequently saved will contain those modifications.

The ldb will automatically be gzipped if the gzip utility from GNU is in the search path. The library database is named **<filename>.gz**. Example:

```
# characterize the library
char_library
# save the library database to tt.ldb
write_ldb tt.ldb
```

write_library

-bus_syntax "<>" | "[ ]" | "( )"
Specifies the bus syntax characters used when outputting the library and the `bus_naming_style` attribute.

-capacitance_only Omit rise/fall_capacitance attributes

-capacitance_range < 0 | 1 | 2 > Output rise/fall_capacitance_range attributes.

-ccs Include CCS data

-ccs_compact Output a compact CCS format data

-ccs_compact_lc Use Library Compiler to output compact ccs.

-ccsn Include CCSN (noise) data

-ccsp Include CCSP (power) data

-ccsp_compact Output compact CCSP format data

-cells {cell_names} List of cell names. Default: *all cells*

-dcnoise_abstol <tolerance>Tolerance used to group similar DC templates. Default: 1e-6 (volts)

-dcnoise_prefix <prefix> Prefix used for writing DC noise templates. Default: "DC_"

-derive_max_capacitance Estimate max_capacitance from transition data.

-driver_waveform Request output of normalized driver waveform.

-driver_waveform_size <value> Number of points in normalized driver waveform. Default: 500

-ecsm Include ECSM data

-ecsmn Include ECSM noise data

-ecsmp Include ECSM power data

-em Include Electromigration data

-exclude Exclude cells given by **-cells**

-expand_buses Turns off the creation of buses and instead outputs individual pins.

-filename <filename> Output file name

-gzip Compress the output library using gzip

-indent <number> Number of spaces to indent by. Default: 2

-map {list} List of name-map pairs, used to map internal names to an external name. Default: `none` (No name mapping.)

-overwrite Overwrite existing .lib file

-precision <precision> Format string to control the precision of the output values Default: "%g"

-preserve_user_data_precision { } List of attributes to have original precision preserved

-rename Rename the existing `.lib` file.

-scan_ccsn_remove Remove ccsn from scan dummy cell

-scan_dummy_scale_power Scale power tables when removing scan pins using the scale factor specified by the **define_cell** command

-scan_output_dummy Convert sequential cells (latches, flops) to scan dummy cells by removing the scan pins

-sdf_cond_equals {"==" | "===" | "== logical" | ""} *sdf_cond* attribute style. Default: "" (none)

-sdf_edges Include `sdf_edges` attribute

-sensitivity_file <filename> When this option is specified, Liberate reads the specified sensitivity file. The sensitivity file is created by 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}	List of data types to be filtered from the output library. Default: none (do not skip any data)
-swap_index_order	Swap the index order for 2D tables.
-sync_ldb	Forces Liberate to read the ldb on disk prior to writing.
-thread <number>	Number of different CPU threads to use. This argument defines the maximum number of threads to use on the current machine. By default, Liberate uses single thread to output library files. Running on two or more threads will save time to output library files.
-unique_pin_data	Keeps unique timing, power, etc. data with each pin bit instead of directly under the bus.
-user_data <filename>	User-provided library data
<libname>	The output library name.

The `write_library` command outputs the library in Liberty format using Tcl API routines. (See ALAPI manual). The Tcl script for formatting the library is provided in the file `$ALTOSHOME/etc/write_library.tcl`.

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 that this can result in a loss in accuracy, as all the data is taken from the first bus bit (the from index).

bus_syntax specifies the bus syntax characters for output to the library, plus the `bus_naming_style` attribute. (See `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. This option is useful for backward compatibility. Care should be taken when using this argument.

capacitance_range requests the output of `rise/fall_capacitance_range` attributes into the library. 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>"
```

cells specifies which cells get written to the output library. If **exclude** is also set then only cells not listed in the `cells` list will be output. By default all cells get written. This option supports the use of a wildcard.

ccs, ccsn, ccsp, ecsm, ecsmn, ecsmp, em, and **si** arguments 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.

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.

Libraries that contain `CCSP` data or `ECSMP` data cannot be created using `write_library` in the same run with other libraries. That is, do not use `write_library -ccsp` or `set_var voltage_map <1|2>` in the same Liberate run with other `write_library` commands. Currently, the following libraries can be written in a single Liberate run: `NLDM` only, `CCS`, `SI`, `ECSM` and/or `CCSN`. The `CCSP` library must be in its own separate Liberate run. Example:

```
read_ldb <ldb>
write_library -ccsp  <ccsp.lib>
```

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.

ccsp_compact outputs a library in the compact CCSP format.

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 variable `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
```

dcnoise_prefix and **dcnoise_abstol** are used when writing DC noise templates. The dcnoise_prefix argument controls the prefix used when naming the templates. The dcnoise_abstol argument controls the merging of DC noise templates. If the noise values are less than this tolerance, then the templates will be merged into a single group.

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 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.

expand_buses specifies the library should be output with individual pins and no buses.

filename specifies the name for Liberate to write out the library. If filename is not specified the library is written to library_name.lib.

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 specifies the number of space to indent. (Default = 2)

map 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 will be overwritten.

precision controls the precision used when writing out the library. The value for this argument must conform to standard Tcl formatting. The default value is "%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, i.e.: myData("1.001") this will be preserved in the output.

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. The `-filename` option specifies the desired output filename. If `-filename` is not specified the library is written to `<library_name>.lib`. The `-gzip` argument compresses the output file using gzip. The`-overwrite` argument disables this automatic version control, and if the output library already exists, it is overwritten.

scan_dummy_scale_power enables power data scaling. The `define_cell` command must include scan related information such as the scan pins for this option to work.

scan_output_dummy removes all scan pins from a cell and writes out the reduced cell.

sdf_cond_equals specifies how `sdf_cond` attributes are written. 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.

skip disables the output of power arcs into the output .lib file. Supported values are: `power`, `hidden_power` and `conditional_hidden_power`**.** 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. Default: use the order specified by the `read_library`, `define_template` or `read_ldb` commands.

sync_ldb forces Liberate to read the ldb on disk prior to writing. Used to address possible precision issues between the `char_library` and `read_ldb/write_library` flows.

user_data file 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.

unique_pin_data 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. (See examples below.)

user_data_override tells `write_library` to allow certain attributes in the `user_data` to override the characterized values.

Examples:

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
```

2:

By default (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) {
           .....
        }
      }
    }
```

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

-abs_tol <value>	Tolerance for comparing templates. Default: 0.0
-auto_index	Generate templates for use with -auto_index option of "char_library"
-cells {cell_names}	List of cell names. Default: *all cells*
-combine_rise_fall_index	Create index_1 and index_2 ranges spanning the maximum/minimum values from both rise and fall arcs
-define_index	Create **define_index** commands for non-default arcs
-dir <directory_name>	Split the template into sub-templates by cell or cell group and store in the named directory. Default: do not split
-driver_waveforms	Converts normalized driver waveforms into define_input_waveform in template.
-exclude	Exclude cells from -cells list
-expand_buses	Generate define_cell commands using **bus_syntax** if buses exist in the input library.
-group <number>	Generate this number of cells per group. Default: all
-group_cells	Generate unique template per cell group; use with the -dir option
-ibis	IBIS template mode
-ibis_char_prefix "string"	IBIS char script file prefix. Default: ibis
-ibis_gen_prefix "string"	IBIS generation script file prefix. Default: igen
-ibis_slew "value"	IBIS input slew. Default: 1n (ns)
-ibis_templ_name "file_name"	IBIS template file name. Default: "template_ibis.tcl"
-index_const "value"	Generate this number of constraint indices for "DATA×REFERENCE". (Default: 3×3)
-index_delay "value"	Generate this number of delay/power indices for "SLEWS×LOADS". (Default: 7×7)
-index_mpw	Generate this number of mpw indices for "SLEWS×LOADS". (Default: 3×2)
-index_si	Generate this number of si_immunity indices for "WIDTHS×LOADS", (Default: -index_delay value)
-input_supply_pin	Output set_pin_gnd / set_pin_gnd.
-io	Create templates with **define_arc** commands for I/O's
-map {list}	List of name-map pairs, used to map internal names to an external name. Default: `none` (No name mapping.)
-merge <"skip_delay">	Generate verbose template with merged "when" conditions.
-mpw	Include mpw templates for each cell

-no_internal_supply	Turns off 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*
-sdf_cond	Write define arcs with -sdf_cond for timing arcs.
-si	Include SI templates for each cell
-skip {leakage}	Do not output `define_leakage` commands.
-sort_pinlist	Liberate will sort the pins in the `define_cell -pinlist` in the template.
-truth_table	Generate templates using read_truth_table input
-unique	Create a unique template for each cell
-unique_power	Generate unique power templates (Default: *use same template indices for delay and power*)
-use_lu_table_name	Reuse the lu_table names from the original library in the Liberate `define_template` commands.
-verbose	Create verbose templates with **define_arc** commands.
-when_as_vector	Converts "when" conditions into a vector sequence rather than having a –when for each **define_arc**.
<filename>	File name for the Tcl template

The **write_template** command creates a Liberate Tcl command file template by reading an existing library (.lib) or library database (ldb). The Tcl command file is created in a file called *filename (.tcl* is appended to the name if it doesn't end in *.tcl*). The Tcl file includes all the necessary **define_template** and **define_cell** commands needed to run Liberate. This function provides a convenient way to use an existing library's templates to create the Tcl file to characterize a new library.

auto_index will generate templates suitable for use with the **auto_index** feature of **char_library**. When **auto_index** is used all cells will refer to a single delay and power template whose size is denoted by the **index_delay** string (default "7×7") and a single constraint template whose size is denoted by the **index_const** string (default "3×3"). If the **si** and **index_si** arguments are set then the cells will refer to a single SI immunity template of size **index_si** (default **index_delay**). The values of the indices in the templates are used as scaling factors for the indices automatically determined from the minimum/maximum transition and minimum load where these are extracted from the input library by **write_template**.

group 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 argument can be useful to generate a list of cells for a trail characterization run with a representative subset of the cells in the library.

group_cells combines cells belonging to the same group into a single file as <group_name>_template.tcl. (See group_attribute and define_group)

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 may then be edited to define index values. The following options will be ignored when using this argument: **cell**, **exclude**, **io**, **unique**, **unique_power group**, and **define_index**. See the Appendix for more information about the truth table syntax.

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 indices and each index is within **abstol** (default 0.0) to each other.

dir specifies a directory where sub-templates for cells or cell groups are stored. File names will be given the names `<dir>/<cell>_template.tcl`. To create unique templates for per cell group, specify the **-dir**, **-group_cells**, and **-unique** options.

If **unique_power** is used, the power templates can have different indices than 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 indices as the delay templates.
Note: using this option can have a significant impact on runtime since additional simulations to characterize the power will be required.

define_index will generate a **define_index** command for each arc of a cell whose indices differ from the default indices 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 indices for different paths within that cell, where the slew and loading conditions used for characterization are different.

driver_waveforms will write out a template that includes `define_input_waveform` commands that can be used to re-characterize 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 **12.1 ISR1** of Liberate 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.

Example showing driver_waveforms in a read_library, write_template flow:

```
read_library my.lib
```

```
write_template -driver_waveforms template.tcl
```

The 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 }
```

The **unique** and **define_index** arguments cannot be used with the **auto_index** argument.

The **io** argument will additionally generate **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 pullup 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 option **verbose** is equivalent to the **io** option with the exception that the pin_load will not be output.

map 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 the user to specify the actual name for the internal pin that needs to be used in the various definitions contained in the template file (`define_cell`, `define_arc`, etc.).

skip disables the output of define_leakage commands into the template file. This option should only be used when "inside_view" is enabled. It cannot be used with the char_library `-io` option since this option disables the inside_view. If there are no define_leakage commands loaded and the inside_view is not enabled, then the resulting library will *not* have any leakage states characterized. Currently the only supported value that can be skipped is "`leakage`".

sort_pinlist option accepts a value of `in_bi_ou`. When enabled, the define_cell -pinlist command option in the template will have the pins sorted as follows: "input bidi output".

si generates **si_iv_curve** and **si_immunity** templates. If the current library does include any signal integrity data then 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 indices are swapped and index_1 actually points to load), the tool recognizes the actual 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 2-D noise immunity template a 2-D template will be created from the delay indices.

The **mpw** argument will generate mpw templates if 2 dimensional mpw tables exist in the loaded library. If 2 dimensional mpw tables do not exist, then no mpw templates will be output.

merge generates a "merged verbose" template with merged "when" conditions. Takes the argument "skip_delay", which only merges `constraint`, `mpw`, and `hidden` power arcs. This must be used together with the -verbose option. Example:

```
write_template -merge skip_delay -verbose
```

Note: When using a "merged verbose" template, you must also set define_arc_preserve_when_string=1.

The **cells** argument controls which cells get written to the template. If **exclude** is also set then cells not listed in the **cells** list will be excluded. By default all cells get written. This option supports the use of a wildcard.

A read_library, read_ldb, read_truth_table or char_library command should be issued before calling write_template.

When generating **define_index** commands (see the verbose, io and define_index options) **write_template** will increase the number of indices to the same size as the default template if it is smaller. For example a pad pin may have a default 7x7 template but for some arcs uses 4x4. As **define_index** requires 7x7, write_template will pad the 4x4 indices by inserting mid-points starting between the 1st and 2nd index, 2nd and 3rd etc until there are enough indices.

Use the option **input_supply_pin** to output **set_pin_gnd** and **set_pin_vdd** commands into the template file. The original library must have pg_pin syntax with related power nodes in the pin group for this feature to work properly.

If buses exist in the input library, then the pins will be output in bus syntax in the **define_cell -input**, **-output**, -**bidi**, **-clock**, -**async** and **–pinlist** arguments. The **–expand_buses** option is used to make **write_template** generate **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] } rather than –output { DOUT [2] DOUT [1] DOUT [0]}.

The bus syntax used for the template can be changed by setting the **bus_syntax** variable. 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> }

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
```

Note: For **ibis_slew**, the user should consult with their design manual to determine an appropriate input slew number to drive the IBIS characterization. A number too large might result in waveform lag when annotating the generated IBIS file into simulation. For example, by comparing IBIS (the B component) annotated waveform with original (direct simulation)

waveform, the following waveform lag might occur. If you see behavior like this, please adjust the input slew number to fix the problem.

write_top_netlist

-cells {list of cells}	List of cells to include in the output (Default: *all cells*)
-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 **write_top_netlist** command creates a Verilog file that contains an instantiation of each cell within the library. The output is written to the given **<filename>**. A **.v** suffix will be added to the filename if it does not end in **.v**. Each instance and pin will be named as follows

```
inst_<cell> pin_<cell>_<pin>
```

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.

The **cells** option can be used to specify a list of cells to be included in the output file. If the **exclude** option is specified, the list of cells specified by the **cells** option will be excluded from the output file.

This command must be used after a database has been loaded.

write_userdata_library

-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 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.

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 the user does not explicitly specify items to include or exclude, the output file will contain 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

-auto_index Automatically generate table indices

-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
-io	Enable IO mode.
<vdb_filename>	Name of the output VDB file

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.

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.

The **write_vdb** command creates a vector library database file for the current library. The VDB file 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 that char_library should not be executed in the same run as write_vdb.

extsim instructs Liberate to use an external SPICE simulator rather than Alspice. Alspice is the default built-in SPICE simulator. The license for the external simulator must be available. Currently, the following external simulators are supported:

- HSPICE
- Spectre
- Eldo
- FineSim

- XA

Important

The setting for **-extsim** must correspond to the setting for the extsim_cmd variable.

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_clients 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.

auto_index instructs Liberate to automatically create the indices 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** indices set the **max_noise_width** parameter.

cells specifies which cells to include in the output.

This command must be used after a database has been loaded.

write_verilog

-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}	**List of cell names. Default: *all cells***
-delayed	Controls naming convention for creating "delayed" signals. Default: "delayed_%P" (where %P is the pin name.)
-exclude	Exclude cells given by `-cells`
-fwire_prefix <prefix>	Prefix for internal wires for pin functions. Default: *int_fwire_*
-indent <number>	Number of characters to indent. Default: *tab*
-merge	Merge cell modules from `user_data`

-mpw_include_output_state
Include the logic state of the output in the condition for mpw checks on "clear" and "preset" signals in the generated Verilog. Default: *don't use*

-mux Use MUX primitives instead of basic logic primitives to represent mux functions. Default: *basic logic primitives*

-notifier Sets the name of the register that is used to flag timing violations. Default: "notifier".

-no_edge Exclude 'posedge' or 'negedge' on edge-triggered arcs. Default is to include edges.

-no_err_primitives Causes `*_err` primitive cells to be replaced with `buf` cells.

-path <path> The string used to denote a path. It must be either "**=>**", or "***>**" Default: "=>"

-sdf_version <version> SDF version, must be **2.1**, or **3.0**. Default: 2.1

-specparams Output delay assignments to *specparams* rather than directly to delay values.

-split_nonunate Instructs Liberate to convert a non_unate timing group in the .lib into 2 arcs in the Verilog output file.

-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>
Adds real timing data into the verilog model using specparams with the specified style. Default: “” (do not use real timing).

-timescale "timescale_value" Change the timescale when writing out Verilog. Default: "1ns/10ps"

-twire_prefix <prefix> Prefix for internal wires of conditional timing constraint functions (Default: *int_twire_*)

-udp_prefix <prefix> Prefix for built-in user defined primitives (UDPs), set to "" to exclude UDPs (Default: *altos_*)

-user_data <filename> User provided data in Verilog format.

<verilog_filename> Name of the output Verilog file

The **write_verilog** command creates a Verilog file for the current library. The Verilog is written to the given **<verilog_filename>**. A **.v** suffix will be added to filenames that do not end in **.v**. The **user_data** argument specifies a user provided Verilog file to merge with the generated Verilog data with. 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. The **write_verilog** command should *not* be used in the same run as the **char_library, read_ldb** or **write_library** commands. Instead, it should be used in a separate Liberate run following a **read_library** command. This is because the Liberty file may have formatting that is required for the Verilog output to be properly formatted. Example:

```
read_library my.lib
write_verilog my.v
```

add_default_udp allows the user to provide user data where not all the primitives are included; default primitives will be used for all other cells. This option only works with `-user_data`.

cells controls which cells get written to the output. If **exclude** is also set, only the cells not listed in the **cells** list will be output. By default, all cells get written. This option supports the use of a wildcard.

delayed 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. Example:

```
write_verilog -delayed "delayed_%P"
```

If there is no "%P" and the `-delayed` string begins with "_" then it is used as a subfix. In the following example, the wire for the pin "CK" is "CK_mysubFix".

```
write_verilog -delayed "_mysubFix" ...
```

if there is no "%P" and the `-delayed` string doe not begin with "_" then 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>**" (delayed_%P) where <pin_name> is a pin that is involved in a timing check.

To turn off generating delayed signals, use "" (empty double quotes). Example:

```
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
```

merge will include cells not specified in the library but present in the **user_data** file.

indent argument specifies the number of spaces to use for indentation, default tab.

mpw_include_output_state option requires the **sdf_cond_style** variable is set to 1. If not set, it will force this setting. Note that this variable should be set prior to creating an equivalent library (.lib) with **write_library**, to ensure consistency between the library and the Verilog. Otherwise, there may be warnings during SDF back-annotation.
This **mpw_include_output_state** option should be used *before* **read_library**, **char_library**, or **read_ldb**. When this variable is used, the mpw check ($width) will only be checked by

Verilog 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 converts pins whose functions are a 2x1 or 4x1 mux into a pre-defined, 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. This option is only needed if **-user_data** is used. (Default: "notifier".)

no_edge excludes 'posedge' or 'negedge' on edge-triggered arcs. This option will change the this:

```
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 don't accept `*_err` primitives.

specparams causes delay assignments in the Verilog to be assigned to *specparam* variables rather than directly to values. The **path** argument controls the delimiter used for delay assignments, either **=>** or ***>**.

sdf_version controls the format of the output Verilog for use with SDF annotation. Set to **3.0** to generate 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.

split_nonunate instructs Liberate to convert a non_unate timing group in the .lib into 2 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 2 delay entries.

table_style requests Liberate to put real timing into the output Verilog file using the specparams syntax. This argument 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 an Standard Delay Format (SDF) file is supplied from the STA tool.

twire_prefix is the prefix used for internal wires created when generating additional functions for state dependent timing constraints. The **fwire_prefix** is the prefix used for internal wires created when generating logic functions. The **udp_prefix** is the prefix used for user defined

primitives that are created for latches and/or flip-flops. Set **udp_prefix** to a null string to exclude generating user defined primitives.

timescale the default timescale is '1ns/10ps'. This option permits changing it to another scale. Example: -timescale "1ps/10fs"

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 (e.g. IQ) to the Verilog state function.
verilog_IQN	- the name map for the internal state of flip-flops (e.g. 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. Must match exactly apart for 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. Must match exactly apart for leading or trailing white space (Default: "*endspecify*")

This command must be used after a database has been loaded. Example:

```
read_library my.lib
# Output a Verilog file
write_verilog -user_data my_verilog my.v
```

write_vital

-cells {cell_names}	List of cell names (Default: *all cells*)
-component <filename>	Filename for component data
-exclude	Exclude cells given by **–cells**
-indent <number>	Number of characters to indent (Default: *tab*)
-merge	Merge cells from **user_data**
-sdf_version <2.1 | 3.0>	Sets the SDF version (Default: 2.1)
-user_data <filename>	User-provided data in Vital format
<vital_filename>	Name of the output Vital (vhd) file

The **write_vital** command creates a Vital VHDL file for the current library. The Vital is written to the given **<vital_filename>**. A **.vhd** suffix will be added to filenames that do not end in **.vhd**. The output will include sequential gates and tristates. The **write_verilog** command should not be used in the same run as the **char_library, read_ldb** or **write_library**

commands. Instead, it should be used in a separate Liberate run following a **read_library** command. This is because the Liberty file may have information that is required for the Vital output to be properly formatted. Example:

```
read_library my.lib
write_vital my.vhd
```

The **cells** argument controls which cells get written to the output. If **exclude** is also set, only the cells not listed in the **cells** list will be output. By default, all cells get written. This option supports the use of a wildcard.

The **component** argument can be used to request printing a list of components into the specified output file. By default no component file is output.

The **merge** option will include cells not specified in the library but present in the **user_data** file.

The **sdf_version** option is used to specify the desired SDF version in the output file.

The **indent** argument specifies the number of spaces to use for indentation, default tab.

The **user_data** argument 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) --
```

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 rather than 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 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`"))

The following Tcl variables can be used to customize the formatting of the generated timing check variables (note these 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):

$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.

This command must be used after a database has been loaded. 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. Liberate-specific parameters are set using the **set_var** command.

adjust_tristate_load

< 0 | 1 | 2 | 21 | 22 > Control if pin capacitance should be added to the load indices on tri-state pins. Default: 1

By default, Liberate will add the pin capacitance of the tri-state pin to each of the load indices when outputting the library. The rise *index_2* will add the rise_capacitance and the fall *index_2* will add the fall_capacitance. In addition when using the **write_template** command to create a Liberate Tcl command file, the tri-state pin rise/fall_capacitance is subtracted from the load indices specified in the input library to create the appropriate **define_template** commands for tri-state pins.

0: Turn off these adjustments; the library and template will *not* add or subtract the tri-state pin capacitance.

1: Add the pin capacitance of the tri-state pin to each of the load indices when outputting the library. (Default)

2: Similar to **1** with the following addition: instead of adding the **rise_capacitance** or **fall_capacitance**, the pin attribute **capacitance** will be added to the load indices for the index_2 values. When using the write_template command, the pin **capacitance** is subtracted from the load indices specified in the input library to create the appropriate **define_template** commands for tri-state pins. The value of the attribute **capacitance** can be modified using the **set_pin_capacitance** command.

This tristate load adjustment can adjust timing arcs, but *not* power arcs. The following settings are supported:

21: Same as **1**, but power arc loads will not be adjusted.
22: Same as **2**, but power arcs will not be adjusted.

This variable can be used after char_library, but must be used before any models are generated. Example:

```
# Disable adjusting tristate pin load indices
set_var adjust_tristate_load 0
```

This variable must be used before **char_library.**

adjust_tristate_load_ccsp

< 0 | 1 >	Controls adjusting the index_2 ccsp dynamic waveforms for tri-state load. Default: 0

Set this variable to 1 for the load indices of the ccsp current waveforms to be modified even for tri-state caps. Note that, when set to the default of 0, modification of load indices in the ccsp current groups is avoided when **adjust_tristate_load** is set to 1 or 2. This variable must be used in conjunction with adjust_tristate_load.

This variable must be used before **write_library.**

alspice_diode

< 0 | 1 >	Controls handling diodes in allspice. Default: 1

Enables an improved algorithm for handling diodes in Alspice. The Default and recommended value is 1. Set this to 0 to achieve the behavior of releases prior to 3.1.

This variable must be used before **read_spice**.

alspice_leakage_option

{options}	Specify Alspice options for leakage simulations.

See also **alspice_option** (below).
This parameter functions the same as alspice_option, except this is targeted to leakage simulations and will override values set by alspice_option. Available options are: **sim_method**, **sim_gmin**, and **sim_step**. (We recommend setting **sim_method** to override the default homotopy.)

Example:

```
# Set the leakage options for Alspice
set_var alspice_leakage_option "sim_method=trap sim_gmin=1e-14 sim_step=1e-14"
```

This variable must be used before **char_library**.

alspice_option

{options}	Specify Alspice simulation control options. Defaults: (No fixed defaults; Alspice determines appropriate values on a case-wise basis.)

This parameter accepts a list of options to be used by Alspice for delay, power and timing constraint characterization. Options are specified in the format "name=value". (See examples below.) Note: This variable has no effect on simulations performed by an external simulator.

Available options:
sim_method: Specifies the integration method. Accepted values are "**trap**", or "**gear**".
Note: We recommend setting this to either **trap** or **gear** 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 user does not specify values for gmin & tstep, Alspice will examine the data and make its own determination about appropriate values. However, Alspice will always use values explicitly specified by the user.

Example:

```
# Set the simulation options for Alspice
set_var alspice_option "sim_method=trap sim_gmin=1e-12 sim_step=1e-12"
```

This variable must be used before **char_library**.

auto_index_distinct_risefall

< 0 | 1 >	Control if the char_library auto_index algorithm will compute both the rising and falling max capacitance values. (Default: ***0***)

Set this variable to allow for distinct rise and fall max_capacitance and load index values when the **char_library -auto_index** option is used.

This variable must be used before **char_library**.

auto_index_input_slew

<value>	Specify the input slew used to calculate `max_capacitance` during `auto_index` characterization.

Set this variable to specify the value (in seconds) of the input slew to be used for all `auto_index` calculations. 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`. The default is to use `max_transition`.

This variable must be used before **char_library**.

auto_index_weak_driver_mode

< 0 | 1 > Controls if the `-auto_index` argument of the **char_library** command adjusts `min_capacitance` of cells that cannot satisfy the `min_transition` at `0` load. Default: `0` (Do not compensate).

Some cells do not have sufficient drive strength to satisfy the `min_slew` condition when not loaded. Setting this variable allows for better correlation between the `min_load` values generated during `-auto_index` between Alspice and `extsim_exclusive` characterizations.

0: Print a warning if `min_slew` does not meet the cell output load. Choose the last attempted valid load. (Default)

1: Same as 0. However, ensure that the methodology is consistent between Alspice and `extsim_exclusive` modes. (Recommended)

This variable must be used before **char_library**.

binning_detail

<low | medium | high> Set detail of state dependency. Default: medium

This parameter is used to set the criteria for determining how individual state -dependent groups will be merged. Specifying high or low detail will result in a progressive increase or reduction in the size of the resulting library. This variable affects the characterization and should be specified before the **char_library** command. Default: medium (Use detailed state dependency).

This variable must be used before **char_library**. Example:

```
# Enable low binning detail
set_var binning_detail low
```

bisection_info

< 0 | 1 | 2 | 3> Set this variable to print additional constraint bisection information. Default: 0

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. The `bisection_info` variable controls the information that is included.

Use this variable for information about debugging constraint bisection searches. To obtain the circuit simulation decks, see the `extsim_deck_dir`, `extsim_save_passed`, `extsim_save_failed` variables, and the `-extsim` argument of the **char_library** command.

It is often easier to follow the debug information while characterizing a single slew or load.

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). 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 a debugging search, this is the recommended setting.

The variable must be set before **char_library**.

bundle_when

< 0 | 1 >	Controls the mapping of bundle pins in a "when" condition. Default: 1

Bundle pins that occur in "when" or "sdf_cond" can either use the bundle name, or a reference bit from the bundle. For example, in the bundle D {D4 D3 D2 D1}

1: Use a "reference" bit from the bundle in the "when". (Default)

```
when : "D4"
```

0: Use the bundle name in the "when".

```
when : "D"
```

Assumes all members have the same direction, i.e.:

```
Possible:
" D1 *  D2"
"!D1 * !D2"

Not possible:
" D1 * !D2"
```

This variable must be used before **char_library**.

bus_syntax

<string> One of "<>", "()", or "[ ]"
Default: "[ ]"

This variable specifies the characters that will be used to delimit bus indices from the bus name. The **bus_syntax** variable 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 the **bus_syntax** variable 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 variable does *not* control the bus_syntax used when writing the output library. That is controlled by **write_library –bus_syntax**.

Any pin in the ldb that ends with a number surrounded by the **bus_syntax** and belongs to a defined bus will be output under a bus group in the Liberty file. Buses are defined either with the **define_bus** command or by the **-input**,
-output, or **-inout** arguments for **define_cell**, using a naming convention of:

```
<bus_name><bus_syntax_open><from_index>:<to_index><bus_syntax_close>"
```

For example:

```
"dout[3:0]"
    or
"din<0:4>"
```

This variable must be used before **write_library**.

capacitance_attr_mode

< 0 | 1 > When set, it separately writes the rise and fall capacitance attributes. Default: 0

This parameter controls when the `rise_capacitance` and `fall_capacitance` attributes should be written separately.

0: Separate `rise_capacitance` and `fall_capacitance` attributes are written only when both capacitances are measured and greater than 0.

1: Separate `rise_capacitance` and `fall_capacitance` attributes are always written.

This variable must be used before **char_library**.

capacitance_force_hidden

< 0 | 1 > Select to output capacitance or a warning.
Default: 0 (Give a warning.)

If when using **char_library -io** and Liberate encounters a hidden arc from a pin that does not have any forward arcs, the following warning will be printed:

Warning (char_library) : Capacitance acquisition is disabled for hidden arc on pin PD_EN.

Set this variable to **1** to enable capacitance on this hidden arc and to disable the warning. Default: **0** (give warning and disable the hidden arc cap).

This variable must be used before **char_library**.

capacitance_pin_rollup_k

<value>	User-defined multiplier used when determining pin capacitance range. Default: 0.5

User-defined multiplier used in formula for calculating pin capacitance range. (See capacitance_pin_rollup_mode) May be set to any value from 0 to 1.0. (Default: 0.5)

This variable must be used before **char_library**.

capacitance_pin_rollup_mode

< 0 | 1 >	Enables an alternate algorithm for calculating pin capacitance and pin capacitance range. Default: 0

Enables an alternate algorithm for rolling up pin capacitance, and determining pin capacitance range.

0: Pin 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. Pin capacitance range **[C_nldm_min, C_nldm_max]** is computed by:

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)]

"K" is a user-defined multiplier set by the variable capacitance_pin_rollup_k. (Default = 0.5)
Note: The K-factor only changes the values in capacitance rise/fall *range* and has no effect on “capacitance” attribute itself.

Note: The **rise_capacitance** and **fall_capacitance** will *still* follow the **set_pin_capacitance** command (similar to mode 0.)

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
```

The variable capacitance_pin_rollup_mode must be used before **char_library**.

capacitance_range_mode

< 0 | 1 > Use absolute min/max or avg of min and avg of max. Default: 0 (Use abs min/max values.)

This variable controls how the capacitance range is calculated.

0: The lower/upper boundary is the min/max value of all the capacitances.
1: The lower boundary value is calculated as an 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 variable must be used before **write_library**.

capacitance_save_mode

< 0 | 1 > Save all pin capacitances. Default: 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.

0: Remove negative capacitances. Default.

1: Save all capacitances. Negative capacitances are saved as positive capacitances (Recommended)

This variable must be set before **char_library**.

ccs_abs_tol

< value > The ccs absolute tolerance. Default: 2.0e-12

Use this control variable to set the CCS absolute tolerance (in seconds). When determining how many points are needed to reproduce the original spice waveform, Liberate will stop adding points to the CCS data when the absolute error between the reduced CCS waveform and the original spice waveform is less than this absolute tolerance.

This variable must be used before **char_library**.

ccs_base_curve_points

< value > The number of base curve points. Default: 10

Use this variable 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 variable can be used after **char_library**.

ccs_base_curve_share_mode

< 0 | 1 | 2 > Change base curve reuse rate. Default: 1

This variable is used to select between different algorithms for reusing CCS base curves. When set to **1** (default), an algorithm using more aggressive base curve re-use rate will be used without impacting accuracy. When set to **2**, a more aggressive CCS compaction algorithm will be used. There will be no significant impact on accuracy when using either mode **1** or **2**. It is recommended to set this variable to **2**. Set this variable to **0** to revert to release 2.3p2 and prior release behavior.

This variable can be used after **char_library**.

ccs_cap_duplicate_risefall

< 0 | 1 > Create a missing rise/fall CCS receiver cap by duplicating a fall/rise CCS receiver cap. Default: 0

If a timing arc has a missing rise or fall CCS receiver capacitance, the default behavior is to copy the appropriate CCS receiver cap from the input pin involved in the timing arc.

1: Create the missing rise/fall CCS receiver cap by duplicating the existing fall/rise CCS receiver cap.
0: Copy the appropriate CCS receiver cap from the input pin involved in the timing arc.

Note: The default method is a more accurate representation of the pin capacitance but may cause problems with downstream tools.

This variable must be used before **char_library**.

ccs_cap_hidden_pin

< 0 | 1 | 2 >	Control the output of pin capacitance for hidden arcs. Default:2 (Output pin cap for hidden arcs)

This variable controls the output of the CCS receiver pin capacitance on inputs with hidden power arcs.

0: For release 2.3p1 and prior release behavior where Liberate only saved the CCS receiver capacitance on "hidden" pins such as the D pin of a flip-flop.

1: Liberate will output 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: Liberate will output CCS receiver capacitance on all input pins that have potential "hidden" conditions; *any* pin that has a hidden power arc will also have CCS receiver capacitance. (Default)

This variable must be used before **char_library**.

ccs_cap_mode_add_missing

< 0 | 1 >	Specifies whether to get 2.4-style libraries with missing receiver caps for one-sided timing arcs. Default: 1

If this variable is set to 0, Liberate will get 2.4-style libraries with missing receiver caps for one-sided timing arcs.

This variable must be used before **write_library**.

ccs_cap_use_input_transition

< 0 | 1 > The ccs receiver cap will follow the input pin direction. Default: 1

Use this variable to instruct Liberate to use the input pin transition direction for CCS receiver capacitance. Previously, the CCS receiver capacitance followed the output direction. To revert to the old behavior where the CCS receiver capacitance follows the output pin direction, set this variable to **0** after read_ldb or char_library and before write_library. The **read_library** command will reset this variable to **0**. The **char_library** and **read_ldb** commands will set this variable to **1**.

If **ccs_cap_use_input_transition** is set to **1** 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) will be use the CCS receiver capacitance from the clock pin. The load index for this entry (index_2) will be set "0" since this data has no load dependency.

This variable must be used after **char_library**.

ccs_cap_use_input_transition_tristate

< 0 | 1 > The ccs receiver cap for tristate pins will follow the input pin direction. Default: 1

Setting this variable to 1 ensures that Liberate always checks that the receiver cap direction follows the input pin with regard to tristate arcs.

This variable must be used before **char_library**.

ccs_force_grid_delay

< 0 | 1 | 2 > Controls accuracy algorithms for checking ccs waveforms. Default: 1

If you don't want to force grid, set this variable to **0**. Set **ccs_force_grid_delay** to **1** to force grid and to check delay accuracy, as well as transition accuracy when segmenting the waveform. Set to 2 to force grid but not to check delay accuracy when segmenting the waveform. The default and recommended setting is **1**.

This variable must be used before **char_library**.

ccs_infer_output_dir

< 0 | 1 > Detect when all outputs have both rising and falling waveforms. Default: 1

Set this to 0 to address **CCST** issues with pulse generators having rising and falling waveforms in every output. (Default: 1).

This variable must be used before **char_library**.

ccs_init_voltage_comp_thresh

< value >	Set a voltage threshold to enable a proprietary initial voltage compensation. Default: 1.1 (fully-compensated)

The Synopsys CCS standard assumes 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 can compensate for the non-rail initial voltage to ensure the CCS waveform matches the NLDM values and reaches the final rail voltage.

This variable represents a percentage of the full-rail swing over which compensation will take 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 variables. If set to a value between 0 and 1, Liberate will compensate to the value given. If set to a value less than 0, compensation is disabled.

Recommended settings:

-1: Disable compensation to improve correlation for PrimeTime 2010.12 and later releases
1.1: Fully enable compensation to improve correlation for PrimeTime 2010.06 and prior releases

Note that 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 will be degraded. See ecsm_measure_output_range for accurate modeling of these effects.

This variable must be set before **char_library**.

ccs_max_current_thresh

< value >	Check for currents greater than the specified threshold. Default: 0.2 (200mA)

Set this variable to a current (in MKS units) to enable a check for large ccs currents that exceed the specified threshold. If the absolute ccs current is larger than this threshold, a warning will be issued.

This variable must be set before **char_library**.

ccs_max_pts

< value > The maximum number of ccs points. Default: 15

Use this control variable to set the maximum number of points that will be allowed in the ccs waveform data. When determining how many points are needed to reproduce the original spice waveform, Liberate will stop adding points to the ccs data when the number of points reaches the limit specified by this parameter. This variable must be set before char_library.

This variable must be used before **char_library**.

ccs_multiple_switching_output_mode

< 0 | 1 > Enables handling of multiple switching output voltage. Default is 0.

When the variable `simultaneous_switch` 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. Note that this could lead to NLDM versus CCS comparison (`compare_ccs_nldm`) failures.

0: (default), 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 variable `measure_target_occurrence`.

This variable must be used before **char_library**.

ccs_rel_tol

< value > The ccs relative tolerance. Default: 0.01 (1%)

Use this control variable to set the CCS relative tolerance. When determining how many points are needed to reproduce the original spice waveform, Liberate will stop adding points to the CCS data when the relative error between the reduced CCS waveform and the original spice waveform is less than this relative tolerance.

This variable must be used before **char_library**.

ccs_segmentation_effort

< 0 | 1 | 2 | 3 | 4 > Controls the CCS waveform acquisition method. Default and recommended value is 1.

This parameter allows selection of the method used to acquire the CCS current waveform from simulated I(t) data.

0: Disable CCS waveform processing. Use values measured from the current waveform.
1: Smoothed algorithm. (Default and recommended)
2: 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 apply CCS retry criteria (ccs_retry_mode) to check the reconstructed waveform instead of the original current waveform.

This variable must be used before **char_library**.

ccs_voltage_smooth_thresh

< voltage | -1 > Set a voltage threshold to enable a proprietary smoothing algorithm. Default: -1 (disabled)

To reduce CCS interpolation error for some corner cases, Liberate can artificially smooth the waveform. Set this to a voltage threshold (e.g. 0.2) to enable the smoothing algorithm. A higher number corresponds more aggressive smoothing. Set this variable to -1 to disable the smoothing.

Note: This algorithm may have an impact on downstream timing or noise tools that are sensitive to that threshold. The default and recommended value is -1.

This variable must be set before **write_library**.

ccs_voltage_tail_tol

< value > The stopping point as a ratio of supply of the CCS waveform for modeling purposes. Default: 0.981

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.

Default: 0.981 (Output must swing to within 1.9% of supply.)
Recommended: 0.951 (Output must swing to within 4.9% of the supply.)

Because 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 exmaple, 5% in Synopsys' Library Compiler).

This variable must be used before **char_library**. For 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`

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`.

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.

ccs_voltage_tail_trim_tol

< value > The stopping point as a ratio of supply of the CCS waveform during simulation waveform capture. Default: 0.98

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.

Default: 0.98 (Output captured to within 2% of supply.)
Recommended: 0.999 (Output captured to within 0.1% of the supply.)

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 will help 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 in time duration would cause problems in legacy downstream tools.

This variable must be used before **char_library**. Example:

```
set_var ccs_voltage_tail_trim_tol 0.999
```

ccsn_active_ccr_recognition_mode

Deprecated. See Backward Compatibility Variables.

ccsn_allow_duplicate_condition

< 0 | 1 > Allows multiple CCSN groups to be written to the library. Default: 0

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 written into the library.

Set to 0: Keep only the worst-case group, and do not include the rest.
Set to 1: Allow multiple CCSN groups to be written to library. (Default)

Note: We recommend setting this variable 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 variable to 0.

This variable must be used before **char_library**.

ccsn_allow_multiple_input_switching

< 0 | 1 | 2> Default: 0

Allows the vectors on the various input pins on a Channel Connected Region (CCR) to switch independently.

0: Use this setting when the input pins to the CCR cannot switch independently.
1: Allow the input pins to the CCR to switch independently. Use this when characterizing CCRs that only propagate noise when the inputs can be toggled in opposite directions independently.
2: This is an enhancement over the setting of 1 to improve the recognition of multiple switching events in CCSN decks.

The recommended value is 0.

This variable must be used before **char_library**.

ccsn_allow_overlap_when

< 0 | 1 > Control output of arc-based ccsn data. Default: 1

Liberate will check for boolean overlap instead of only an explicit *when* string comparisons. This improves on an issue in Liberate where vector data is not properly matched up to the corresponding timing groups for arc based ccsn generation. This occurs only in the write_template verbose flow because the timing group can have a non-empty *when* string, but the Vecdata would have a null string for the *when* condition. Default: 1, which allows checking for boolean overlap. Set this to 0 to revert back to the release 2.3 and earlier behavior.

This variable must be used before **char_library**.

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

By default, Liberate will use 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 will issue a Warning during characterization and a Warning during modeling. These CCSN groups will not be written out in the library.

Set to 1: Liberate will override the recommendations with the voltage levels measured in the transition. Where possible, the voltage levels from the recommendations will be used. If necessary, indexes and values will be padded to insure monotonicity.

This parameter must be used before **char_library**.

ccsn_arc_channel_check

< 0 | 1 > Enable aggressive pruning. Default: 1

Enables an aggressive pruning of non channel connected ccsn stages that might end up as arc-level ccsn constructs by enabling a more thorough vector dependent check.

Set to 1: Enables pruning. (Default and recommended setting.)
Set to 0: Disables pruning. (Achieves 3.0p3 and prior behavior.)

This variable must be set before **char_library**.

ccsn_arc_consistent_cut

< 0 | 1 > Enable an enhanced CCS cut algorithm. Default: 1

The CCSN generation in Liberate uses an enhanced algorithm to ensure that a consistent cut node is chosen for arc based ccsn. Liberate will choose a single stage CCSN arc over a two

stage CCSN arc whenever possible. Set this variable to **0** to revert back to release 2.4p2 and prior behavior.

This variable must be set before **char_library**.

ccsn_arc_high_effort

< 0 | 1 > Control output of arc-based ccsn data. Default: 1

Liberate outputs more arc-based ccsn stages by applying "looser" side input requirements. The result is that some types of cells, such as muxes, will have an increase in the number of arc-based ccsn data. This control variable can be used to revert back to the pre 2.2p2 behavior. Acceptable settings are:

0: Revert to the pre 2.2p2 behavior, which does not output state-dependent ccsn arcs.
1: Enable the state dependent CCSN arcs. (Default)
2: Same as 1, also outputs a message into the log file for each cell that triggers state-dependent arc code.

This variable must be used before **char_library**. 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: 2

0: Do not model unbuffered outputs. (Behavior of release 3.2p2 or earlier.) Not recommended.
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. (Default and recommended.)

This variable must be used before **char_library**.

ccsn_channel_inputs_high_effort

< 0 | 1 > Control merging of CCSN data. Default: 1

This variable was introduced to enable CCSN constructs for un-buffered input pins. Set this variable to 0 to revert back to v2.3 and earlier behavior.

This variable must be used before **char_library**.

ccsn_compatibility_mode

Deprecated. See Backward Compatibility Variables.

ccsn_consistent_side_inputs

< 0 | 1 > Controls merging of ccsn data. Default: 1

This variable is enabled to avoid a crash that can occur when there are CP->Q arcs with no **when** conditions but different internal side-input values. Reset this variable to a 0 to revert to release 2.3 and earlier behavior.

This variable must be used before **char_library**.

ccsn_dc_static_check

< 0 | 2 | 3 > Performs a check to determine if the CCSN partition is valid. Default: 0

0: Do not perform any data checks on CCSN DC tables. (Default)
1: *(Deprecated; this setting is not used.)*
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 magnitide difference of 10 or greater to be considered a valid CCSN partition. For more information, see ccsn_dc_static_check_thresh.

This variable must be set prior to **write_library**.

ccsn_dc_static_check_mode

< 0 | 1 > Controls severity of messaging when a problem in the CCSN data it 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. (Default)

This variable must be set before **write_library**.

ccsn_dc_static_check_thresh

<value> Default: 10

This variable specifies the threshold applied when `ccsn_dc_static_check=3`. Set this to the desired threshold that will identify a turn on or a non-static behavior in a CCSN DC table.

This variable must be set before **write_library**.

ccsn_dc_template_size

<value> Specifies the ccsn DC current table size to be used.
Default: 29

This variable is used to specify the size of the DC current table. For larger geometries, this variable 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, the recommended setting is 29.

This variable must be used before **char_library**.

ccsn_default_group_add_when

< 0 | 1 > Adds WHEN condition inside default ccsn groups.
Default: 0 (Do not add WHEN.)

Set to **1**: this will add WHEN conditions inside the default ccsn groups.
Set to **0**: this will not add WHEN conditions inside the default ccsn groups. (Default)

This variable must be used before **char_library**.

ccsn_default_group_criteria_mode

< 0 | 1 > Enables generation of worst case ccsn default groups at the pin level and in the default timing group. Default: 0

Set this variable to get the worst case CCSN default groups at the pin and at the default timing group. This variable 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.

1: Generate the most conservative per-corner libraries.
0: Generate libraries that are structurally independent of per-corner data. (Default)
Important: For libraries to be used in voltage or temperature scaling flows, we highly recommended setting this parameter to 0.

This parameter may be used after **char_library**.

ccsn_dual_tie_enable

< 0 | 1 > Enable CCSN modeling for tieHI and tieLO cells. Default: 1

Set this variable to **0** to revert back to 2.5 and prior release behavior, where ccsn data was not output for tiehi/lo cells.

This variable must be used before **char_library**.

ccsn_extra_default_stages

< value > Control merging of ccsn data. Default: 1

When this parameter is set to 2, Liberate will try to add redundant default CCSN stages on all cell pins.

This variable must be used after **char_library**.

ccsn_fanout_select_mode

Deprecated. See Backward Compatibility Variables.

ccsn_input_xfr_probe_mode

Deprecated. See Backward Compatibility Variables.

ccsn_io_mode

< 0 | 1 > Enables ccsn model generation when **define_cell –type** is set to UDA or when **char_library -io** is used. Default: `1`

This variable must be used before **char_library**.

ccsn_io_mode_enable

< 0 | 1 > Enables IO CCSN circuit partitioning algorithm. Default: 1

The generation of CCSN data requires special circuit partitions. When the Liberate Inside View algorithm is enabled, the circuit partitions are constructed from the characterized timing arcs. When the `char_library -io` mode is used, Inside View is disabled. In this case, the

CCSN circuit partitions are constructed from the circuit by tracing the CCC (Channel Connected Component/Region) connected to the cell port.

0: Do not use the IO CCSN circuit partitioning algorithm when Inside View is enabled.

1: Allow Liberate to use the IO CCSN circuit partitioning algorithm if the Inside View-based partitioning fails to successfully characterize the CCSN. (Default and Recommended)

This variable must be used before **char_library**.

ccsn_model_unbuffered_output

Deprecated. See Backward Compatibility Variables.

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 variable must be used before **char_library**.

ccsn_pin_criteria_mode

< 0 | 1 > Adds additional criteria for modeling CCSN groups as pin-based instead of arc-based. Default: 1 (pin-based)

Multiple criteria may be considered when deciding whether to model a stage as pin-based or arc-based. Setting this variable adds additional criteria that will push a stage to the pin.

0: Favor arc-based modeling. Should only be used for backward-compatiblity purposes to match 12.1.5 or prior releases.
1: Favor pin-based modeling (Default and recommended)

This variable must be used before char_library.

ccsn_pin_high_effort

< 0 | 1 > Select a ccs effort algorithm. Default: 1

Liberate can create CCSN stages for 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. Set this variable to **0** to revert to release 2.4p2 and prior behavior. The default (and recommended) setting is **1**.

This variable must be used before **char_library**.

ccsn_pin_stage_merge_mode

< 0 | 1 | 2 | 3 > Control merging of ccsn data. Default: 3

It is often desirable to merge CCSN stage_type pull_up and pull_down data to stage_type "both". Use this variable to instruct Liberate to prefer ccsn_stage_both for input/output probe nodes that use inverter input/outputs.

Set this variable to enable enhanced conditional checks to understand "don't care" conditions. Previously, before Liberate considered two groups to be equivalent (which allowed Liberate to merge them together into 1 group.) Liberate required that 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.

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, will choose inverter input whenever possible. (Default and Recommended setting.)

This variable must be used before **char_library**. Example:

```
set_var ccsn_pin_stage_merge_mode 0
```

ccsn_pin_unconditional

Deprecated. See Backward Compatibility Variables.

ccsn_pin_voltage_level_attrib

< 0 | 1 | 2> Writes the Liberty attributes input_signal_level and output_signal_level into CCSN groups. Default: 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. (Recommended)
2: Update all CCSN stages (arc and pin) with the CCSN signal attributes regardless of the multi-domain status of the cell.

This variable must be used before **char_library**.

ccsn_prefer_min_vt_probe

< 0 | 1 > Methodology for determining CCB output probe point selection. Default: 0 (Recommended is 1)

Use this variable when a CCSN stage (Channel Connected Region) has more than one output that can be selected as the output probe. When this variable is set to 1 Liberate will select 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 will maximize 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.

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. (Default)

1: (Noise-on-delay model) Select probe node that will result in a full output swing (Has no Vt drop from power rail), if possible. (Recommended)

This variable must be used before **char_library**.

ccsn_prefer_two_sided_stages

Deprecated. See Backward Compatibility Variables.

ccsn_prop_noise_peak_mode

< 0 | 1 > Used with CCSN characterization of level shifters. Default: 0

Set this for CCSN characterization of level shifters if Library Compiler issues warnings about incorrect index_1 voltage ranges.

0: (Default)
1: Set this to control the propagated noise input values.

ccsn_prop_retry_duration_incr

Deprecated. See Backward Compatibility Variables.

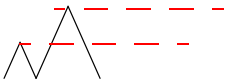
ccsn_prop_retry_peak_incr

<value> Controls the input PWL generation for propagated noise.

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:

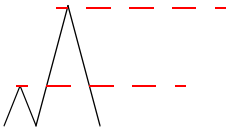
Given vdd=1, gnd=0, and initial peak=0.1

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

ccsn_probe_mode

Deprecated. See Backward Compatibility Variables.

ccsn_prune_last_stage

< 0 | 1 > Set to prefer two-sided stages. Default: 1

Liberate will skip **ccsn_last_stage** generation on some output pins. When enabled, this variable will cause Liberate to skip generation of all pin based ccsn stages for output pins that are not destinations of timing() arcs and will output two sided (stage_type: both) ccsn stages whenever possible. Reset this variable to **0** for compatibility to release 2.3 and prior behavior where the last stage is not skipped.

This variable must be used before **char_library**.

ccsn_xfr_ccc_probe_mode

< 0 | 1 > Disables probing of transmission gate outputs when modeling ccsn first-stage groups. Default: 1

This variable ensures that, when modeling ccsn first-stage groups, Liberate will not probe transmission gate outputs when the transmission gate input was the arc pin. Set **ccsn_xfr_ccc_probe_mode** to 0 to go back to release 3.0 and prior behavior.

This variable must be used before **char_library**.

ccsp_base_curve_points

< value > The maximum number of points in a ccsp base curve. Default: 10

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 will use up to the specified number of points in a shared base curve.

This variable must be used before **write_library**.

ccsp_default_group

<min | max | off> Enable writing a ccsp default group. Default: off

This variable is used to enable the output of a default CCSP power group. A value of **min** selects the dynamic_current which has the min internal_power. A value of **max** selects the dynamic_current which has the max internal_power. The default is **off** which means no default CCSP group will be output.

This parameter may be used after **char_library**. Example:

```
# Enable a max default ccsp power group
set_var ccsp_default_group max
```

ccsp_leakage_current_abstol

<value> Set the ccsp leakage current absolute tolerance. Default: 1.0e-10

Use this control variable to set an absolute tolerance of when Liberate will adjust ccsp leakage current for Library Compiler compliance. Default: 1.0e-10. This default conforms to the requirements in Library Compiler 2008-09 SP1. For older releases of Library Compiler such as 2007-12 SP1, use a value of 0.

This variable must be used before **char_library**.

ccsp_leakage_current_compensation_mode

< 0 | 1 >	Methodology used for ccsp_leakage_current_abstol compensation. Default: 0

This variable controls the methodology used for compensating **leakage_current** values for Library Compiler compliance.

0: Adjust the rail with smallest leakage_current. (Default)
1: Adjust a 0v rail. If multiple 0v rails are attached to a cell, then adjust the primary_ground with the *most* leakage_current. (Recommended)

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 variable:

```
set_units -timing 1ps -capacitance 1ff -current 1ua -leakage_power 1pw
```

Also: This set_units change usually has the added benefit of decreased file size with the same (or slightly improved) accuracy.

This variable must be used before **write_library**.

ccsp_min_pts

<value>	Set the minimum number of CCSP samples. Default: 6

Set this variable to the minimum number of desired ccsp samples. This variable will only have an effect when the variable **ccsp_segmentation_effort** is enabled.

This variable must be used before **char_library**.

ccsp_pin_direction_post_default

< 0 | 1 >	Controls finding unateness of ccsp dynamic groups. Default: 0

When set to 1, when determining the pin direction (rise|fall) of the CCSP power group, Liberate will examine 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 **char_library**.

ccsp_prune_factor

<value> Set the CCSP tail decay factor. Default: 2

Use this variable to control the decay to be applied after the ccsp_prune_second_tol. If this variable is set to **2** (default), the ccsp waveform will be pruned at ccsp_prune_factor*(ccsp_prune_start_tol - ccsp_prune_second_tol) after ccsp_prune_second_tol. See **ccsp_prune_start_tol** for more information.

This variable must be used before **char_library**.

ccsp_prune_second_tol

<ratio> Set the second CCSP tail prune tolerance. Default: 0.03 (3%)

Use this variable to control the second point used when pruning the CCSP waveform tail. See the variable **ccsp_prune_start_tol** for more information.

This variable must be set before **char_library**.

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 will terminate the CCSP waveform early. Liberate will follow 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 will then follow 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 will terminate the waveform after a time 2x (see **ccsp_prune_factor**) from the **ccsp_prune_second_tol** point.

When this variable is set to **0** (default), no waveform trimming is done. Note that **ccsp_prune_second_tol** must be < **ccsp_prune_start_tol**. If this check fails, Liberate will automatically adjust **ccsp_prune_second_tol** to 0.5 * **ccsp_prune_start_tol**.

Setting this variable to the desired value (i.e.: 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 will be 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 will also check 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 variable must be used before **char_library**.

ccsp_quantization_num_steps

<value> Set the number of CCSP quantization steps. Default: 100

Set this variable to the number of quantization steps used to preprocess the supply current waveform to smooth out kinks.

This variable must be used before **char_library**.

ccsp_rel_tol

<ratio> Set the CCSP relative tolerance. Default: 0.02

Set this variable to the tolerance of the area segmentation current as compared against the original area of the supply current.

This variable must be used before **char_library**.

ccsp_related_pin_mode

< 0 | 1 > Enable combining of dynamic current groups. Default: 0

Liberate will write out separate dynamic current groups for each output pin. This can give wrong power reports when using CCSP format data. Set this variable to 1 to enable combining the dynamic current groups under each output pin into one common group whenever possible. The recommended setting when using CCSP is 1.

This variable must be used before **char_library.**

ccsp_segmentation_effort

< 0 | 1 | 2 | 3 > Choose the CCSP segmentation algorithm. Default: 3

Set this variable to enable a more accurate CCSP segmentation algorithm. Setting **ccsp_segmentation_effort** to 0, disables segmentation for supply current waveforms. If set to 1, Liberate segments the supply current waveform and minimizes the number of points. When set to 3, **ccsp_segmentation_effort** 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. Do not use 2, as this is currently just for engineering use. The recommended setting is 3.

This variable must be used before **char_library**.

ccsn_simultaneous_switch_save_vecdata

< 0 | 1 > Enables generation and saving of `vecdata` for select pins. Default: 0 in 12.1.5; 1 in 13.1 and later

Use this variable to enable the generation and saving of `vecdata` for inputs such as the select pin(s) of a mux when `simultaneous_switch_from_cell_when=1`. The `vecdata` is required when generating CCSN data for an arc in a library.

0: CCSN 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 Release 12.1 ISR4 and prior. (Default)

1: Generate and save all `vecdata` for `select` pin even when `simultaneous_switch_from_cell_when=1`. (Recommended)

This variable must be set prior to **char_library**.

ccsp_table_reduction

< 0 | 1 | 2 > Enable reduced CCSP table data. Default: 1

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.

0: Disables table size reduction.
1: Use a 3x3 table for CCSP data. (Default)
2: Enables a 2x2 table.

This variable must be used before **char_library**

ccsp_tail_tol

<value> Set the CCSP tail tolerance. Default: 0.05

Set this variable to the percentage of area considered as the waveform "tail". The tails are discarded during the preprocessing of CCSP waveforms.

This variable must be used before **char_library**.

cell_port_case

<upper | lower | preserve> Provides control for how all cell pins are output.
Default: preserve

Set **cell_port_case** to upper or lower to convert cell pin names to all upper or all lower case in the output ldb and library. The default is preserve, which 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 variable must be used before **read_spice**.

cell_use_both_ff_latch_groups

< 0 | 1 | 2 > Enables support for multiple *ff* and *latch* groups in the output library.
Default: 0

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 variable to enable support for modeling multiple function groups.

When set to 0 (default), if the **write_library** user_data file contains both *ff* and *latch* groups, Liberate includes one of the latch groups in the output library. This setting is provided for backward compatibility.

When set to 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.

When set to 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.

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 variable must be set before **write_library**.

char_mos_term_cap

< 0 | 1 > Sets the method for estimating transistor device pin capacitance. Default: 0 (Biased method)

This variable is used to enable 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.

0: Enables the biased method. (Default)
1: Enables the integration method, which is used automatically if `extsim_exclusive=1`. (Recommended)

This variable must be set prior to **char_library**.

cleanup_tmpdir

< 0 | 1 > Allow Liberate to delete temporary data directories. Default: 1

When using external simulators, Liberate will create temporary directories and store data files in them (example: altos.<pid>.<index>). When this variable is set to 1, Liberate will attempt to remove these directories when it exits. If Liberate exits due to a simulation error or when terminated by the host system, or if this variable is set to 0, then the temporary directories will not be removed.

This variable must be used before **char_library.**

combinational_out_to_out_arc

< all | all_timing | buf | none > Controls the stage-types that Inside View considers when determining output-to-output arcs. Default: "buf".

"**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. (Recommended.)
"**buf**": Limits checking for output-to-output arc to single input gates between the two outputs, such as a buffer or inverter. (Default.)
"**none**": Output-to-output arc is disabled, except for user defined ones using define_arc.

The default value is "buf". The recommended value is "all_timing".

This must be set before **char_library**.

combinational_risefall

< 0 | 1 > Enable writing arcs of timing-type *combinational_rise/fall* for preset/clear pins. Default: 0 (off)

This variable is used to enable *combinational_rise* or *combinational_fall* arcs on preset/clear-related pins. By default, arcs with a related_pin of preset/clear will not have *combinational_fall/rise* timing groups output.

This parameter may be used after **char_library**. Example:

```
# Enable combinational arcs for preset/clear
set_var combinational_risefall 1
```

conditional_arc

< 0 | 1 > Control conditional arcs. Default: 1 (output conditional arcs)

Set this parameter to **0** to turn off the storing of conditional arcs in the library database. All states will be characterized using a proprietary algorithm which will result in a faster runtime but only one state being stored in the ldb. This will reduce the size of the ldb and will also result in all output libraries being generated with no state dependency. Default: **1** (store arcs with full state dependency). When set to **0**, this variable will override **conditional_constraint**.

This variable must be used before **char_library**.

conditional_cap_hidden_pin

< 0 | 1 > Default: 0

Liberate can 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.

0: Liberate will output unconditional (state-independent) CCS/ECSM `receiver_cap` for each input pin if the capacitance data was characterized and stored. The variable `ccs_cap_hidden_pin` or `ecsm_cap_hidden_pin` determines if the capacitance is characterized and saved for the hidden arcs.

1: Liberate characterizes and saves all conditional (state-dependent) pin-based receiver capacitance associated with each characterized hidden power arc.

This variable must be set before **char_library**.

conditional_cap_hidden_pin_thresh

<value> Default: 2

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.

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.

The maximum supported value is 64.

This variable must be used before **char_library**.

conditional_constraint

< 0 | 1 | 2 > Control conditional constraint characterization for timing constraints. Default: 0 (off)

This parameter is used to enable characterization of conditional timing constraints including setup, hold, recovery, removal and minimum pulse width. By default, timing constraints are characterized under worst-case conditions. Setting this variable to **1** will characterize each unique *when* condition (note this will increase run-time). If set to **2** the worst-case constraint condition is characterized similarly to **0,** but a combined *when* condition is generated in the library. If conditional_arc is set to 0, this variable will affect characterization, but only one state independent timing group will be stored. This variable should be set before char_library.

This variable must be used before **char_library**. Example:

```
# Enable conditional arcs for timing constraints
set_var conditional_constraint 1
```

conditional_expression

<merge | separate> Control whether conditional groups are merged or not when they contain sub-expressions that are OR'd together.
Default: merge

This parameter is used to control whether the conditions containing OR sub-expressions are separated into different groups or not. The default is to **merge** all conditions that result in identical values by ORing each of the sub-expressions. Set to **separate** to split into distinct

groups. Also, when **conditional_expression** is set to **separate**, default timing and default power groups will also be created if enabled by the **default_timing** and **default_power** control variables.

This variable must be used before **char_library**. Example:

```
# Split groups with OR'd conditions
set_var conditional_expression separate
```

conditional_hidden_power

< 0 | 1 > Control state dependence of hidden power.
Default: 1 (output state-dependent hidden power)

When set to a 0, only unconditional hidden power will be output. Default= 1 (output only state dependent hidden power.)

This variable must be used before **char_library**. Example:

```
# Turn off conditional states for hidden power
set_var conditional_hidden_power 0
```

conditional_immunity

< 0 | 1 > Control NIC (noise immunity curve) data. Default: 0

This parameter is used to enable characterization of conditional noise immunity curve (NIC) data. By default, the worst case NIC will be output. Setting this variable to **1** will request the output of unique NICs for each *when* condition (Note: this will increase run-time).

This variable must be used before **char_library**. Example:

```
# Turn on conditional states for noise immunity
set_var conditional_immunity 1
```

conditional_include_constant

< 0 | 1 > Control whether constant nets are included in conditional *when* statements. Default: 1

This parameter is used to control whether or not a constant pin will be included in the conditional *when* statement. By default, constant pins will be included in conditional *when* statements.

This variable must be used before **char_library**. Example:

```
# Disable constants in 'when' conditions
set_var conditional_include_constant 0
```

conditional_include_output

< 0 | 1 > Include the output pin in the *when* condition of hidden power and leakage constructs. Default: 1 (on)

This parameter controls whether the *when* condition of hidden power and leakage groups can include the output pin or not. Default is to include the output pin.

This variable must be used before **char_library**. Example:

```
# no output pins in hidden power & leakage 'when' conditions
set_var conditional_include_output 0
```

conditional_leakage

< 0 | 1 > Control state dependence of leakage. Default: 1 (output state dependent leakage)

Use this variable to disable state dependent leakage. Default: 1 (output state dependent leakage).

This variable can be used after **char_library**. Example:

```
# Disable conditional states for leakage
set_var conditional_leakage 0
```

conditional_mpw

< 0 | 1 > Control state dependence of mpw. Default: -1

Use this variable to control the output of conditional mpw timing constraints. By default, mpw constraints will follow the conditional_constraint setting. The supported values are:

-1: Follow conditional_constraint (default)
0: Turn off conditional mpw, regardless of conditional_constraint
1: Turn on conditional mpw, regardless of conditional_constraint

This variable must be used before **char_library**.

constraint_async_probe_internal

< 0 | 1 > Specifies whether to probe internal nodes on non-sequential constraints. Default: 1 (permits probing internal nodes).

Liberate automatically finds probe nodes for measuring constraints. Use this variable to allow Liberate to select internal nodes as the constraint probe node. This variable only applies to `non_seq_setup/hold` constraints. (See also constraint_probe_internal.)

0: Prevents Liberate from choosing an internal node as the probe node for non-sequential constraint acquisition.
1: Permits Liberate to probe internal nodes. (Default)

This variable must be set before **char_library**.

constraint_bisection_mode

< 0 | 1 | 2 > Methodology controlling bisection searches. Default: 2 (Updated Brent's Method)

This variable allows users to control the methodology behind Liberate's bisection searches. Prior to version 3.2p2, only setting 1 was available.

0: Use pure bisection. Result will be correct.
1: Use Brent's Method. Could be artificially pessimistic when compared to pure bisection in some corner cases. (Should only be used for backward-compatibility)
2: Use improved Brent's Method. Result should be within 1 constraint tolerance of pure bisection, but use fewer iterations. (Default and recommended)

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 1 or 2 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 should be 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 variable must be used before **char_library**.

constraint_check_final_state

< 0 | 1 | 2 | 3 > Enable check for the final transition state. Default: 0 (Do not check the final state.)

This parameter enforces additional criteria on constraint measurements.

The final state of internal probe node and feedback wires will be checked to have fully transitioned, and the final voltage on each of the internal probe and feedback wire must agree with their intended logic level.

0: Do not check the final state. (Default)

1: The circuit simulation will continue until each of the internal probe and feedback wires reach to within 5% of their intended voltage. This setting ensures that the wires reach 95% of their voltage swing.

2: The simulation will not stop early. This setting ensures that the output transition has fully settled before measuring the degradation.

3: Same as option 2, but also check for stability of internal feedback wire at the end of the simulation. If any internal feedback voltage is within or below `constraint_check_final_state_threshold`, the iteration will be considered as failing. (Recommended)

Note: A setting of 2 or 3 may increase a small number of constraint values and will slightly increase characterization time.

This parameter normally only affects constraints with delay degradation as the measurement criteria. A setting of 3 may also affects constraints with glitch-peak criteria, as glitches larger than the `constraint_check_final_state_threshold` will be considered as failing measurements.

This variable must be used before **char_library**.

constraint_check_final_state_threshold

<value>	Set the final state threshold. Default: -1 (use delay_out_fall / rise)

Set this variable to the desired final threshold to be used when **constraint_check_final_state** is enabled. The threshold value is a ratio between **0** and **1**. When allowed to default, the threshold will use the **delay_out_rise** threshold for a rising transition and the **delay_out_fall** threshold for a falling transition.

This variable must be used before **char_library**.

constraint_check_rebound

< 0 | 1 >	Check if the output rebounds. Default: 0

When measuring constraints using delay criteria as the metric, Liberate expects that the probe node will transition from gnd to vdd monotonically. It does not check if the probe node reverses direction. Sometimes the probe node transition reverses course momentarily. When this variable is set to `1`, if a probe node transition reversal crosses the delay measurement threshold (see delay_out_fall and delay_out_rise) multiple times, then Liberate measures the time delta between the first measurement threshold crossing and the last. If the delay delta between these 2 crossings exceeds 1pS, then this bisection bound is treated as a failing bound. When set to the default value of `0`, Liberate does not check for probe node reversals that lead to multiple measurements.

This variable must be used before **char_library**.

constraint_clock_gater

<value>	Set to 0 to disable special clock gater circuit constraints. Default: 1 (enable special checks)

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. Default: **1** (clock gater constraints have special handling). To disable special handling, set this variable to a **0**.

This variable must be used before **char_library**.

constraint_combinational

<value>	Set to 0 to disable combinational measurement of constraints. Default: 0

When set to 0 (default), Liberate will use a bisection search method when characterizing constraints. When set to 1, Liberate will first attempt to use a bisection search, but when that fails, it will use a simple delay-degradation based sweep in an attempt to characterize the constraint. Setting this variable to 2 enables a path-difference, formula-based constraint estimation if the bisection search fails to characterize the constraint. To characterize a constraint for a combinational cell such as a nand gate, this variable should be set to 1. Note that a setting of 1 or 2 can significantly slow down the Liberate run.

This variable must be used before **char_library**.

constraint_combinational_step_limit

<value>	Specify the number of sweep steps. Default: 1000

When constraint_combinational is enabled (set to 1 or 2), this variable will determine how many sweep steps the constraint characterization can take..

This variable must be used before **char_library**.

constraint_combinational_step_size

<value>	Set to the desired combinational constraint stepsize. Default: 10e-12 Seconds

Use this parameter to specify the step size (in MKS units) to be used for linear constraint searches (non-bisection, see constraint_combinational).

This variable must be used before **char_library**.

constraint_delay_degrade

<value>	Percentage of delay degradation used when characterizing timing constraints (setup, hold, recovery, removal). Default: 0.1 (10%)

This parameter is used to specify the maximum amount of delay degradation 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.

The set_constraint_criteria command can also be used to set this variable. If both this variable and the **set_constraint_criteria** are used, the last one executed will set the value to be used by Liberate.

This variable must be used before **char_library**.

constraint_delay_degrade_abstol

<delay>	Minimum delay degradation value (in seconds) used in characterizing timing constraints (setup, hold, recovery, removal). Default: 5e-12 (5 ps)

This parameter is used to specify the minimum delay degradation value permitted (in seconds) in the clock-to-constraint output pin (flip-flop) or the data-to-`constraint_output_pin` (latch) delay. 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.

Note: For **28nm and below**, or for high performance designs we recommend a value of 2e-12.

The set_constraint_criteria command can also be used to set this variable. If both this variable and the `set_constraint_criteria` are used, the last one executed will set the value to be used by Liberate.

This variable must be used before **char_library**.

constraint_delay_degrade_abstol_max

<delay>	Maximum delay degradation value (in seconds) used in characterizing timing constraints (setup, hold, recovery, removal). Default: -1 (not used)

This parameter is used to specify the maximum delay degradation value (in seconds) permitted in the clock-to-constraint output pin (flip-flop) or the data-to-constraint_output_pin (latch) delay. 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.

This variable must be used before **char_library**. 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 >	Set this to minimize the D to Q delay. Default: 0 (Do not minimize D to Q delay)

Set this variable to use the minimum <setup | recovery> + clk-to-Q delay as the setup and recovery criteria instead of the **constraint_delay_degrade** criteria. Notice that D does not propagate to Q, but rather, "minimizing D-to-Q" is shorthand for minimizing the sum of D->clk and clk->Q. *(Also, see note below.)*

0: Use the constraint_delay_degrade criteria when measuring setup and recovery. (Default)

1: The setup measurement threshold for degradation cases will be 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 will all be 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/fall_constrained_timing group.

3: The same as setting of 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).

Note: When this variable is set to a value other than 0 or 1, the clk to Q delay measurement will be simulated with both D and clk switching with the time offset equal to the setup time derived previously.

This variable must be used before **char_library**.

constraint_delay_degrade_minimize_dtoq_clock_only

< 0 | 1 > Limits `minimize_dtoq algorithm` to clocked arcs (the related_pin is a clock). Default: `0` (apply to all constraint arcs).

This variable controls whether the constraint_delay_degrade_minimize_dtoq algorithm will be applied only to clocked constraints (where related_pin is a clock) or to all constraints.

0: When the constraint_delay_degrade_minimize_dtoq algorithm has been 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 variable must be set before **char_library**.

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

This variable 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 really 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.

0: use whatever setup time is available. (Default)
1: constrain delay to probe only
2: constrain delay with worst setup data only
3: 1 and 2
4: constrain delay for non-scan setup only
5: 1 and 4
6: 2 and 4
7: 1, 2 and 4

This variable must be set before **char_library**.

constraint_delay_degrade_minimize_dtoq_tol

< 0 | 1 >	Specifies a tolerance on the minimum dtoq delay. Default: 0 (Do not apply the tolerance)

Set this variable to specify a tolerance on the minimum dtoq delay so that setup and recovery times can be reduced, while increasing clock to q delay, so that the total dtoq delay is:

```
(1+constraint_delay_degrade_minimize_dtoq_tol) * min_dtoq
```

This variable must be used before **char_library**.

constraint_delay_min_check

<min_time>	Minimum valid reference delay for constraint acquisition Default: -1 (seconds)

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 (e.g. 50%), the reference delay may be close to 0 or even negative.

Liberate will check the reference delay against the value of **constraint_delay_min_check**, and report an informational message if the reference delay is smaller.

The default value of constraint_delay_min_check is -1 (seconds), which effectively disables the check.

This variable must be used before **char_library**.

constraint_dependent_recrem

< 0 | 1 | 2 > Enable re-characterization of recovery (1) or removal (2). Default: 0 (standard characterization)

This variable provides separate control over dependent characterization for recovery and removal timing constraints. See **constraint_dependent_setuphold** for more detailed information.

This variable must be used before **char_library**.

constraint_dependent_setuphold

< 0 | 1 | 2 > Enable re-characterization of setup (1) or hold (2). Default: 0 (standard characterization)

Use this variable to enable dependent setup and hold characterization. A value of **1** requests re-characterization of setup. A value of **2** requests re-characterization of hold. To re-characterize for dependent setup and hold, an ldb must be loaded using read_ldb that includes standard setup and hold data. Default = 0 (standard setup and hold characterization). An example of **2** is shown in the **Figure 4** below.

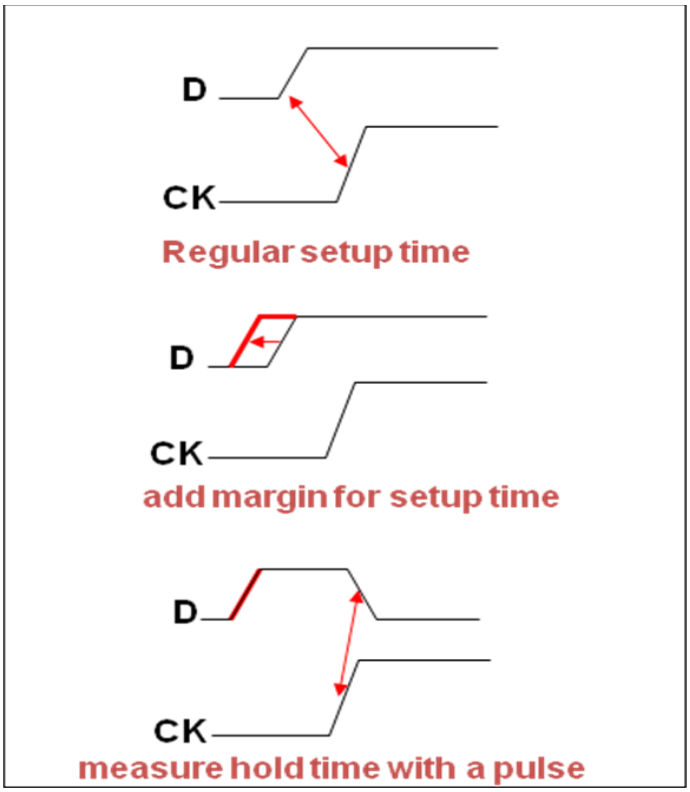


Figure 4: Dependent Hold time

This variable must be used before **char_library**.

constraint_dependent_setuphold_input_threshold

<value> Specify minimum input peak threshold. Default: -1

When dependent setup/hold is enabled the data input will be 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. Use this variable to control the minimum height of the input triangular waveform. The acceptable value is a ratio of the supply voltage and is between 0 and 1. Default: -1 which means to use the value of the control variables delay_inp_rise and delay_inp_fall depending on the input transition direction.

This variable must be used before **char_library**.

Example:

```
# Set the dependent setuphold input threshold
set_var constraint_dependent_setuphold_input_threshold 0.7
```

constraint_dependent_setuphold_margin

<value> Specify a setup or hold margin to be applied during dependent setuphold characterization. Default: 0 (seconds)

Use this variable to add a margin to the hold (setup) time when re-characterizing for dependent setup (hold) constraints. Default: 0 seconds.

This variable must be used before **char_library**.

constraint_dependent_setuphold_margin_ratio

<ratio> Specify Enable re-characterization of setup (1) or hold (2). Default: 0 (standard characterization)

Use this variable to add a margin to the hold (setup) time when re-characterizing for dependent setup (hold) constraints. The total margin is determined using the formula specified below. Default: 0

This variable must be used before **char_library**.

<dependent_margin> + <dependent_ratio> * (index_1 + index_2)

constraint_dependent_setuphold_pessimism

< 0 | 1 > Specify a setup or hold margin to be applied during dependent setuphold characterization. Default: 0 seconds

Use this variable to guarantee pessimism over independent setup/hold values. When set to **1**, Liberate will report the more pessimistic value between the dependent and independent characterized constraint values. Default: 0 (always use dependent characterized constraint value).

This variable must be used before **char_library**. Example:

```
# Set the output pin name for constraints
set_var constraint_dependent_setuphold_pessimism 1
```

constraint_failed_value

<value> Inserts the value of this variable into the ldb and resulting library. Default: 1 (second)

If a constraint characterization fails (like when the result is too close to search bounds), Liberate inserts the value of the **constraint_failed_value** into the ldb and resulting library. The default is 1; for example, 1e9 in library units of ns. Set this variable to **-1** to revert to the 2.5 (and prior) behavior, where if – for example – 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 some rare cases, this could be a large negative value, which would result in an optimistic constraint value in the library.

This variable must be used before **char_library**.

constraint_glitch_hold

< 0 | 1 > Use glitch height for determining hold constraints. Default: 0 (use delay degradation)

This parameter is used to enable glitch-height as the failure criteria for characterizing hold-time constraints, rather than delay degradation. In the default method, hold measurements are made by measuring the delay degradation of the output switching to its failure state. This results in a more pessimistic hold constraint time than probing for a glitch in the output waveform to signify failure.

This variable must be used before **char_library**.

constraint_glitch_peak

<value> Glitch height as a ratio of supply used in characterizing timing constraints (setup, hold, recovery, removal). Default: 0.1 (10%)

This parameter is used to specify the maximum size of logic glitch permitted on the constraint output pin before an arriving signal is deemed to fail a timing constraint.

The set_constraint_criteria command can also be used to set this variable. If both this variable and the **set_constraint_criteria** are used, the last one executed will set the value to be used by Liberate.

This variable must be used before **char_library**.

constraint_glitch_peak_internal

<value> Glitch height as a ratio of supply used in characterizing timing constraints (setup, hold, recovery, removal).
Default: follow `constraint_glitch_peak`.

This variable specifies the maximum size of logic glitch permitted on an internal node before an arriving signal is deemed to fail a timing constraint check. The default is to follow `constraint_glitch_peak` (or removal_glitch_peak for removal arcs).

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.

The recommended setting is the default, which follows `constraint_glitch_peak` (or `removal_glitch_peak` for removal arcs). See `constraint_glitch_peak_mode=2` for recommendations for clearing most "Too close to search bounds" errors.

This variable must be used before **char_library**.

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 this variable times VDD, then the threshold will be limited to the voltage represented by the ratio of VDD specified by this variable. This can result in a search bound error.

If there is a rail-to-rail intrinsic glitch (i.e. always present) then this can be disabled by setting it to a value greater than 1.0 but less than 2.0.

This variable must be used before **char_library**.

constraint_glitch_peak_mode

< 0 | 1 | 2 > Apply `constraint_glitch_peak` on top of inherent glitch. Default: 0 (Recommended: 1)

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 will contain 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.

0: Do not apply `constraint_glitch_peak` on top of inherent glitch. (Default)

1: Liberate will measure 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 will be 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. (Recommended)

2: Operates the same as option 1, but only pertains to constraint arcs that probe internal nodes (such as clock gaters). If a constraint arc probes an internal node, it operates the same as option 1. However, if a constraint arc probes an external pin, it operates the same as option 0. This mode may be useful when the inherent glitch on internal nodes are larger than the inherent glitch on external pins.

This variable must be used before **char_library**.

constraint_glitch_peak_report_inherent

< 0 | 1 > Reports in the log file the measured inherent glitch on the probe node. Default: 0

This parameter is used to enable the reporting of the measured inherent glitch on the probe node. 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 related pin. It is measured by Liberate only when `constraint_glitch_peak_mode` is set to `1` or `2`. This glitch is reported in the log

file when enabled by this variable. The default is `0`, which means that Liberate will not report the measured inherent glitch.

This variable must be set before **char_library**.

constraint_hold_probe

<value>	Name of cell probe pin to use for hold timing constraint characterization. Default: "" (follow the parameter constraint_output_pin)

This parameter is used to specify the name of the node in the cell used for determining hold values. The **value** can also be an internal node specified using the form **<transistor_name>:[S|D|G]** were **S** is the transistor source, **D** the transistor drain and **G** the transistor gate terminal.

This variable must be used before **char_library**. Example:

```
# Set the output pin name for constraints
set_var constraint_hold_probe ProbePin
```

constraint_info

< 0 | 1 | 2>	Enable printing of constraint measurement details. Default: `0` (don't print)

Setting this variable to `1` will turn on printing of constraint mapping information (including the probe point) into the log file. The information printed will include the cell name, timing type, constraint type, pin, related pin, probe node, and criteria.

Setting this variable to `2` will turn on printing of constraint mapping information after a cell is finished with characterization. Constraint criteria may be updated during simulation. (Recommended)

This variable must be used before **char_library**. Example:

```
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>	Default: 1

This variable is used to control the print out of glitch metric as "glitch" or "pass/fail".

0: Print glitch metric as "glitch"
1: Print glitch metric as "pass/fail" (Default).

This variable must be set before the **char_library** command.

constraint_linear_waveform

< 0 | 1 > Set to "1" to request a linear waveform for constraint characterization.
Default: 0 (use same driver that delay and transition use)

This parameter is used to request that a linear waveform be used to drive the inputs during constraint characterization. By default, the input pins will use the same driver as the delay and transition characterization.

This variable must be used before **char_library**.

constraint_margin

<value> Set to a add a margin to combinational constraints.
Default: 2e-12 (seconds)

Use this control variable to specify a margin (in MKS units) to apply when the combinational constraint method is used.

This variable can be used after **char_library**.

constraint_merge_state

< 0 | 1 > Set to enable merging of timing groups. Default: 1

Liberate will merge timing groups if 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 and/or probes. Set this variable to **1** to enable Liberate to merge these arcs using the bitwise worst case among the values.

This command must be used before **char_library**.

To revert to release 2.4 and prior behavior, set this variable to **0**.

constraint_output_load

<min | max | value | index_?> Type of loading to use when characterizing constraints.
Default: min

This parameter is used to set the type of load on the output pin used for constraint characterization. The options are:

min: The minimum load index_value (Default)

max: The maximum load index value

value: Specify an exact load to use

index_? : Specify 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" where the load index contains 7 points.

This variable must be used before **char_library**. 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>	Name of output pin to use for timing constraint characterization Default: "" (use alphabetically first output pin name)

This parameter is used to specify the name of the output pin used for determining setup, hold, recovery and removal values. The **pin** can also be an internal node specified using the form **<transistor_name>:[S|D|G]** were **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 will be reported. Using this functionality can result in approximately a 12% increase in run-time, depending on the number of constraints to be characterized.

This variable must be used before **char_library**. 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 method for selection of constraint and MPW probe nodes. Default: 0

Use this variable to control the selection method for constraint and MPW probe nodes when using the `-io` mode option of the **char_library** command, if the **define_arc** command does not specify the desired probe node. In the `-io` mode option, it is recommended to specify the desired probe node.

0: (default) Issues an error.
1: Determines the probe node by checking the setting of the `constraint_output_pin` variable. 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 is the recommended setting.

This variable must be set before **char_library**.

constraint_probe_internal

< 0 | 1 > Specifies whether to probe internal nodes for constraints. Default: 1 (permit probing internal nodes).

Liberate's Inside View automatically finds probe nodes for measuring constraints. These probe nodes can be primary cell ports or can be internal nodes. This variable applies to all constraints.

0: Prevents Liberate from choosing an internal node as the probe node for constraint acquisition.
1: Permit Liberate to probe internal nodes. (Default)

This variable must be set before **char_library**.

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 methodology for selecting internal probe nodes for constraints. Default: 0 (Only select probes from back-to-back CCCs).

The Liberate Inside View (patented) algorithm 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. Inside View has several criteria for judging whether or not a node is suitable to be a probe point.

This variable applies to all constraints.

0: Only select probes from back-to-back CCCs. (Default)
1: Select probes from any internal simwire that connects to a transistor gate.
2: Bugfix for mode 0 (Recommended)
3: Bugfix for mode 1.

The recommendation is to use mode 2. If Liberate fails to find an appropriate probe node in mode 2, which results in all constraints of a given type being skipped, then use mode 3.

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 variable. 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 variable must be set before **char_library**. For example:

```
set_var constraint_probe_mode 2
```

constraint_probe_upper_fall

<value>	Probe upper fall threshold. Default: 0.7

constraint_probe_upper_rise

<value>	Probe upper rise threshold. Default: 0.7

Use these variables to set the voltage thresholds to be used when measuring clock and data delays. See the *define_arc -pin_probe_threshold* and **-related_probe_threshold** for more information.

This variable must be used before **char_library**.

constraint_search_bound

<min_time> Set this variable to the minimum desired search bound. Default: -1

Use this variable to set the initial constraint search bounds. The constraint bisection search will start at a maximum of +/- the search bound value. By default, Liberate automatically determines the search bound for optimal runtime. When this variable is set to a time (in seconds), Liberate will start the bisection search using +/- the constraint_search_bound value which will override the automatically determined bound value.

This variable must be used before **char_library**.

constraint_search_bound_bisection_mode

< 0 | 1 > Specifies bisection initial bound algorithm. Default: 0

The bisection search requires an initial passing search bound and an initial failing search bound. Use this variable to enable to determine the definition of a pass and a fail.

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 will switch (passing search bound) when data arrives well in front of clock, and will fail 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 will always switch and the delay pushout will determine the pass/fail for the bisection search.

Note: This mode will be applicable if delay pushout is the criteria. This is because the glitch criteria should already work with the bisection algorithm.

constraint_search_bound_estimation_mode

< 0 | 1 | 2 > Controls how Liberate determines the constraint search bound Default: 2

This variable controls the method used to determine the constraint search bound. It is recommended not to change this parameter except for backwards-compatibility purposes.

0: Use the method prior to release 3.1.
1: Use the method prior to release 12.1.4.
2: Use a method that is consistent for all settings of extsim_model_include and extsim_exclusive. (Default and recommended)

This variable must be set before **char_library**.

constraint_search_bound_probe_mode

< 0 | 1 >	Set this to expand probe selection. Default: 0

Liberate supports any node that is an output of a channel or the input to a gate to be used as a probe node. When this variable is set to 1, then Liberate will also allow additional nodes such as input nodes connected to pass transistors.

This variable must be used before **char_library**.

constraint_search_time_abstol

<min_time>	Set this variable to the minimum desired search time. Default: 4e-12 (seconds)

Set this control variable to the minimum constraint binary search window. This variable will affect characterization and should be set prior to the **char_library** command. This variable can have a significant impact on runtime if set too small. Default: 4e-12 seconds

This variable must be used before **char_library**.

constraint_slew_degrade

<value>	Include slew degradation percentage along with delay degradation as a criteria for calculating timing constraints. Default: -1 (only use delay degradation)

This parameter is used to include slew degradation when determining timing constraints. By setting the slew criteria both delay degradation and slew degradation will be checked and the first criteria to fail will determine 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_*** variables.

The set_constraint_criteria command can also be used to set this variable. If both this variable and the set_constraint_criteria are used, the last one executed will set the value to be used by Liberate.

This variable must be used before **char_library**. 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 >	Set this variable to enable the constraint characterization algorithm to snap to the passing bound. Default: 0 (Use Cadence proprietary algorithm.)

Set to 1: Forces Liberate to snap to the passing bound while characterizing constraints when processing a binary search. By default, Liberate uses a proprietary algorithm to determine the final constraint value. Default: 0.

The recommended setting for this parameter is 1. This may result in a bit more pessimism and runtime.

This variable must be used before **char_library**.

constraint_tran_end_mode

< 0 | 1 | 2>	The methodology for setting transient simulation end time in constraint bisection searches. Default: 1(Optimized per table entry).

SPICE simulators have different capabilities and limitations. This variable affects how the transient simulation end time is set when using various simulators.

0: Use mode 2 for Spectre, SKI, and Titan; otherwise use mode 1. (Backward compatible only)
1: Use an optimized tran_tend (transient simulation end time) for each entry of each iteration except for Titan, which uses mode 2. (Default and recommended)
2: Use the maximum tran_tend from each entry in the constraint data table for all simulations. This mode should only be used for testing purposes.

This variable must be used before **char_library**.

constraint_user_defined_probe_mode

< 0 | 1 > Use only user-defined probes for setup/hold, recovery/removal. Default: 0

1: Liberate will use only user-defined probes for setup, hold, recovery and removal. (Note: this is already the behavior for MPW.) User-defined probes are specified with either define_arc -probe or set_constraint_criteria -probe.
0: Behavior of release 12.1.1 and earlier where Liberate treats the user-defined probes as a suggestion. (Default)

This variable must be set before **char_library**.

constraint_vector_equivalence_mode

< 0 | 1 > Enables a vector equivalency check that helps reduce the number of vectors simulated. Default: 0 (off)

Selects an algorithm that determines which vectors need to be simulated for constraints and which can be skipped as equivalent to the simulated vectors. 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 would require 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.

0: Use base algorithm for reducing vectors based on matching all switching modes. (Default)
1: Enables a vector equivalency check that helps reduce the number of vectors simulated. (Recommended)

This variable must be set before **char_library**.

constraint_vector_mode

< 0 | 1 | 2 > Mode for controlling recovery, removal, and mpw vectors when using define_arc. Default: 0

0: Liberate will use the pre-3.0p3 level of effort for vector generation. (Default)

1: If a **define_arc** is supplied for recovery, removal and/or min_pulse_width types, Liberate increases the vector generation effort.

2: This mode specifies the probing of internal nodes for all types of constraints/mpw when default probing fails to find a vector. This provides more automation in a recharacterization flow (especially for PMK) where certain arcs cannot be acquired without specifying the internal probe node. Examples include MPW on SAVE pin, and non_seq constraints between RDN and RETN pins. (Recommended)

This variable must be set before **char_library**.

constraint_width_degrade

<value>	Percent pulse width degradation. Default: 0.1 (10%)

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. Used together with the `-metric width` option of `define_arc` command.

constraint_width_degrade_abstol

<value>	Minimum width degradation value (in seconds) used in characterizing timing constraints (setup, hold, recovery, removal) for pulse latch cells. Default: 5e-12 (5ps)

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 constraints 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.

constraint_width_degrade_abstol_max

<value>	Maximum width degradation value (in seconds) used in characterizing timing constraints (setup, hold, recovery, removal). Default: -1 (not used)

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 constraints 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. Both `constraint_width_degrade_abstol` and `constraint_width_degrade_abstol_max` can be used simultaneously to set both an upper and a lower bound to the delay degradation.

constraint_worst_vector_abstol

<min_time>	Set this variable to control minimum constraint binning. Default: 0 (do not apply an absolute tolerance)

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 variable to the same value as **constraint_search_time_abstol**.

This variable must be used before **char_library**.

debug_flow

Provides for a quick debug flow. The user can check selected points in the slew/load matrix by shrinking the template to a 2×2, or 1×1:

A **2×2** matrix will use the first and last indexes.
A **1×1** matrix will use just the first index.

This is specified with a "set" command (not a set_var). Example:

```
...
set debug_flow 2x2
...
```

This variable must be set before the first **define_template** command.

def_arc_drive_side_bidi

< 0 | 1 >	Set this variable to drive bidi pins that are not the pin or related_pin of an arc. Default: 1

When this variable is set to 1, Liberate will drive any bidirectional pin that is not the pin or the related_pin of an arc. Set this variable to 0 to tell Liberate not to drive side pins that are bidirectional.

This variable must be set before **char_library**.

def_arc_msg_level

< 0 | 1 >	Instructs Liberate to report define_arc commands that will not be characterized. Default: 1

0: Liberate will report an error when no valid vectors for the arc are found.
1: Will report a warning for every define_arc command that will *not* be characterized. (Default)

Note: The write_template command is enhanced to add "set_var def_arc_msg_level 0" This change will ensure that all new template files created from write_template will follow the behavior prior to release 3.2.

This variable must be set before **char_library**.

def_arc_vector_consistency_check

< 0 | 1 >	Switch to specify the "when" state must agree with the pre-switching state of the -vector. Default: 1

1: The "when" state must agree with the pre-switching state of the -vector. (Default)
0: Set the behavior of release 3.1p4 and earlier.

This variable must be set before **char_library**.

default_power_avg_mode

< 0 | 1 >	Use a weighted average or a simple average. Default: 0 (simple average)

Set this variable to enable a weighted average of all internal_power states under a pin. 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 actual groups under the pin will be used. When this variable is set to **1**, the default *internal_power* will be computed as a weighted average of all of the available states. That is, the default_power will be computed as though **conditional_expression** was set to **separate**.

This variable must be used before **write_library**.

define_arc_ignore_mode

< 0 | 1 | 2>	Controls whether Inside View considers the "when" condition for "define_arc -ignore" commands. Default: 0

Inside View will use the information given in the define_cell command as a starting point for determining a cell's functionality. Liberate's default behavior is to find all vectors possible from this general description.

In cases where the design intent is different from the actual circuit behavior, there are additional controls to restrict the vectors that Inside View generates. The "define_cell" command accepts a "-when" option which, when used with "set_var

simultaneous_switch_from_cell_when 1" will restrict the full vector set to only vectors that satisfy the "when" condition.

This variable allows similar fine-grained control for the define_arc command. When define_arc_ignore_mode is set to 1 and a "define_arc -ignore" command is given with a -when option, Liberate will ignore all arcs that satisfy that "when" condition.

0: If the `-ignore` argument of the **define_arc** command is used, Liberate ignores all arcs for the pin and related_pin. All other **define_arc** options are also ignored. This is the default setting.
1: In addition to 0, Liberate considers "when" condition in the define_arc command.
2: In addition to 1, Liberate considers the "type" condition in the define_arc command.

This variable must be used before **char_library**.

define_arc_preserve_when_string

< 0 | 1 >	Set this variable to preserve a "when" string and output to the library. Default: 0 (off)

Set this variable to preserve a "when" string in its original form and output to the library. Normally Liberate will attempt to simplify a "when" condition according to Boolean logic. (For example, in the following command: `define_arc -when "A&B | A&!B"`, Liberate could simplify the "when" condition to `-when "A"`.)

0: Allow Liberate to simplify the "when" string. (Default)
1: Preseve the "when" string in its original form and output to library.

define_duplicate_cap_mode

< 0 | 1 >	Specifies the aliasing of pins for the determination of pin capacitance. Default: 0 (off)

This works together with the define_duplicate_pins command to specify which pin will be used to characterized the pin capacitance, and then applied to the "duplicate" (aliased) pins.

0: No pin aliasing for pin cap characterizations; each pin cap will call for a separate simulation to characterize its value. (Default)
1: Use aliasing to determine pin capacitance; a given pin will be characterized, and its value will be applied to all the other pins specified with the **define_duplicate_pins** command.

This variable must be used before **char_library**.

delay_constrained_by_setup_recovery

< 0 | 1 > Controls how timing constraints will be characterized. Default: 0

When the **delay_constrained_by_setup_recovery** variable is set to 1, the **constraint_delay_degrade_minimize_dtoq** variable is forced to 0. Liberate will characterize the timing constraints with the normal delay degradation method, but clock to output delays will be measured with the data pin also switching – at a time offset as determined by the measured setup/recovery time. The recommended setting for **delay_constrained_by_setup_recovery** is 0.

This variable must be used before **char_library**.

delay_inp_fall

<value> The % point on the cell input falling waveform to measure delays from. Default: 0.5 (50% of supply)

delay_inp_rise

<value> The % point on the cell input rising waveform to measure delays from. Default: 0.5 (50% of supply)

delay_out_fall

<value> The % point on the cell output falling waveform to measure delays to. Default: 0.5 (50% of supply)

delay_out_rise

<value> The % point on the cell output rising waveform to measure delays to. 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_rise**, **delay_inp_fall**, **delay_out_rise** and **delay_out_fall**) 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 variables must be used before **char_library**. Examples:

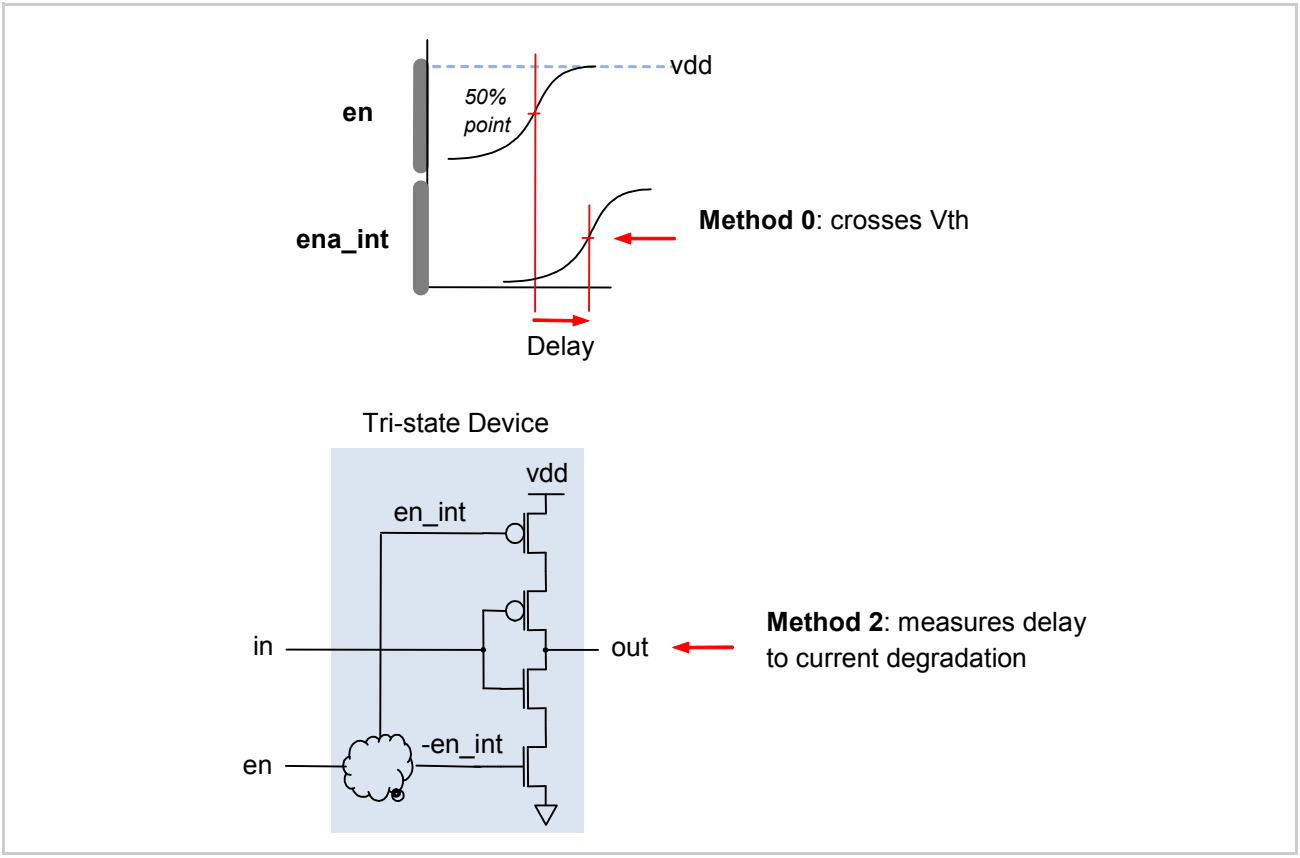
```
# Set the delay measurement from 45% to 55%
```

```
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 *three_state_disable* arcs (disable timing) are calculated in tri-state devices. Default: 0

This variable controls how the timing arcs for tri-state devices are calculate when the device is transitioning to the OFF state (three_state_disable.)



Set to 0: 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) **crossing the threshold voltage** Vth. (Default)

Set to 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).

Set to 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 will respect that regardless of the disable_method.

This variable must be used before **char_library**. 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`.)

See timing_group_unateness.

0: Does not remove `timing_sense` (Default)
1: Removes the `timing_sense` after the merging of two timing groups.

This variable must be set before **char_library**.

disk_wait_time

<value>	Controls how long Liberate will wait before attempting to read the sim.lis file. Default: 0 (seconds)

When Liberate is using an external simulator (see char_library -extsim), Liberate will attempt to read the simulator output file (sim.lis for Hspice) as soon as the simulator terminates. Due to network latencies, the sim.lis file may not yet be ready to read. Set this variable to the number of seconds you want Liberate to wait before it attempts to read the external simulator output file.

This variable must be used before **char_library**.

driver_cell_acc_mode

< 0 | 1 >	Generates a more accurate driver waveform for minimum slews. Default: 1 (on)

0: Reverts to 3.1p2 (or earlier) behavior in sampling active driver waveforms.
1: Generates a more accurate driver waveform for the minimum slews. (Default)

This variable must be used before **char_library**.

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 will be held to a constant value by the specified driver cell.

1: Same as setting '0' except all constant side pins will be driven by the specified driver cell.

This variable must be used before **char_library**.

driver_cell_info

< 0 | 1 > Enables reporting of driver cell by pin. Default: 0

Set this variable to request a report for each cell input. The report will include the driver cell that will be the characterized driver for the input. This report is useful to determine the actual driver cell assigned to each input when many set_driver_cell commands are used.

This variable must be used before **char_library**.

driver_cell_load_all_outputs

< 0 | 1 > Enables reporting of driver cell by pin.
Default: 1 (apply the load to the secondary side pins)

Use this variable to control if the active driver load will be 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.

This variable must be used before **char_library**.

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.
Default: 0

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. The **set_driver_cell** commands can be executed before the **write_library** command or if the commands are stored in the ldb (see **enable_command_history**), this variable enables Liberate to honor them.

This variable must be used before **read_ldb**.

driver_waveform_arcs_only

< 0 | 1 > Enables to choose when to save driver waveforms. Default: 0

Use this variable to select the method Liberate will use to filter the input driver waveforms that will be stored and available for output into the .lib file. Set this variable to a **1** to store only those input waveforms used to characterize arcs for delay and power. When set to **1**, if there are no delay or power arcs, then no input driver waveform data will be stored. When the library (or pin) is characterized only for input capacitance (see char_library **-skip**), the library database will not have any characterization data from delay or power tables to save. When set to **0** (default), Liberate will write 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. Set this variable to **1** to revert back to the release 2.4p2 and prior release behavior.

This variable must be set before **write_library**.

driver_waveform_pulse_mode

< 0 | 1 > Enables to choose the pulse algorithm. Default: 1

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. When this variable is set to **1**, Liberate will create a new point to approximate where the two waveforms cross.

To revert to release 2.4 and prior behavior, set this variable to **0**.

This variable must be used before **char_library**.

driver_waveform_wildcard_mode

< 0 | 1 > Controls formatting of the Normalized Driver Waveform (NDW) data in the output library. Default: 0

When the define_input_waveform command is used to specify the input waveforms and the pin names specified with it contain wildcards, this variable can be used to enable proper formatting of the NDW data in the output library.

0: Wildcards in the **define_input_waveform** commands are not supported when writing NDW data in the output library. If wildcards exist in a **define_input_waveform** command, NDW will not be written out.

1: Wildcards in the **define_input_waveform** commands are expanded when writing NDW data into the output library. This can lead to many large NDW tables in the output library.

The recommended value is 0. In addition, it is recommended that wildcards are not used in the **define_input_waveform** command.

This variable must be used before **char_library**.

duplicate_pin_attr_mode

<use_master | augment_master> Controls how master pin attributes will be populated to duplicated pins.

use_master: All attributes from the master pin will be copied to the duplicate pin except those generated from characterization, ie: pin capacitance. (Default)
augment_master: Copy all the attributes from the master pin to duplicated pin, but don't overwrite attributes which have already been set for that pin.

duplicate_risefall_power

< 0 | 1 > Enable 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.

When this variable is set to 1, and only rise or fall power (but not both) exist, duplicate the power values. For example, if only *rise_power* exists but not *fall_power*, then copy the *rise_power* values into *fall_power*. This duplication will only apply to *internal_power* groups where the output pin is switching. It will not apply to hidden power. The default is 0, where the missing power direction is represented by a scalar group with value of 0.

This variable can be used after **char_library**. Example:

```
# Duplicate existing power to missing power
set_var duplicate_risefall_power 1
```

duplicate_risefall_power_ccsp

< 0 | 1 | 2 > Controls copying rise/fall ccsp dynamic currents into a missing rise/fall group. Default: 0

If this variable is set to 1, Liberate copies rise/fall ccsp dynamic currents into the missing rise/fall group (similar to what the **duplicate_risefall_power** variable does). Setting **this variable** to 2 does the same as setting it to 1, except Liberate includes the input_switching_condition in the copied group. If set to 0, rise/fall ccsp dynamic currents are not copied.

This variable must be used before **char_library.**

ecsm_cap_hidden_pin

< 0 | 1 | 2 > Specifies how ECSM capacitance is written for input pins. Default: 2

Prior to version 3.1 the variable **ccs_cap_hidden_pin** (default = 1) controlled how the ECSM capacitance was written. The variable **ecsm_cap_hidden_pin** 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. (Default)

This variable must be used before **write_library.**

ecsm_cap_input_slew_mode

< 0 | 1 > ECSM capacitance input slew adjustment mode. Default: 0

0: Write the original index_1 for ecsm_capacitance. In this mode, the index_1 slew values in ECSM data tables will not be adjusted. (Default and recommended.)
1: Use the adjusted index_1 slew values from release 3.1p2 and earlier. *This adjustment is no longer recommended.*

This variable must be used before **write_library**.

ecsm_cap_mode

< 0 | 1 > Set the ECSM cap model mode. Default: 1 (Output a 1 or 2-piece cap table.)

Set this variable to enable the output of a capacitance table with 3 steps. Liberate will characterize and write out the capacitance data into the ldb. The three capacitance steps are fixed at the levels specified by the control variables **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: Output a 1 or 2 piece cap table.
1: Output a 3-piece cap table. Dependent on ecsm_version (Default)

Note: To output a 3-piece cap table for input pins, we also recommend setting the variable ccs_cap_hidden_pin to 2.

This variable must be used before **write_library.**

ecsm_cap_use_input_transition

< 0 | 1 > Sets ecsm_capacitance to follow the input transition.
Default: 1 (Follow the input.)

0: ecsm_capacitance will follow the output transition.
1: ecsm_capacitance tables follow the input transition. (Default & Recommended.)

This must be set before **write_library**.

ecsm_invert_gnd_current

< 0 | 1 > Inverts current values for ecsm_current_waveform groups.
Default: 1 (Invert current.)

1: Inverts the current values (index_1) for ecsm_current_waveform groups associated with ground supplies. (Currents will be normally positive instead of normally negative.) This is for compatibility with EPS. (Default)
0: Do not invert currents.

This variable must be set before **char_library**.

ecsm_measure_output_range

< 0 | 1 > Enable scaling of the voltage span for the ECSM format.
Default: 0 (Do not scale).

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 (example: Vss to Vdd).

0: Provides backward compatibility to the 3.2 and prior releases. (Default)
1: Liberate will modify 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
```

Note: 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 will be more characterization failures on those designs, unless this parameter is set to 1.

In addition, the following ECSM attributes will be 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 variable to 1. If this is supported, we recommend a setting of 1.

This variable must be set before **char_library**.

ecsm_version

< 1.1 | 2.0 | 2.1 > Set the ECSM version. Default: 2.1

This parameter specifies the ECSM format version. Supported versions are: 1.1, 2.0, and 2.1

Version 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.

Version 2.1: Supports an ecsm_capacitance with 2-piece or 3-piece capacitance tables, dependent on the setting of ecsm_cap_mode.

If no advanced multi-piece receiver capacitance data is stored in the ldb, then a single capacitance value is output (per ECSM 2.0).

This variable can be used after **char_library**. Example:

```
# Set the ECSM version to 2.1
set_var ecsm_version 2.1
```

ecsmn_mode

< 0 | 1 > Controls ECSMN noise modeling behavior for pin-level or arc-level constructs. Default: 0

Set this variable to 1 to generate arc-based ECSM Noise models for single-stage channel-connected circuits (CCC). This can improve accuracy at smaller geometries, and is recommended by Cadence.

0: Use pin-based ECSM-N models for all groups (Default).

1: Use arc-based ECSM-N models for single-stage CCCs and pin-based models for 2+ stages (Recommended).

This variable must be used before **char_library.**

em_calculation_monitor_rails

< 0 | 1 > Enables EM calculation on power supply rails. Default: `1`

Specify if the EM calculation should be performed on the power supply rails. Runtime can be improved by skipping the rails.

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. Default

For more information, see the Liberate Details section on Electromigration.

This variable must be used before **char_library**.

em_clock_freq

<**value**> The EM clock frequency in Hertz. Default: “”

Specify the clock frequency at which to characterize electromigration. The value specified must be a number in Hertz. The scientific notation is supported. For more information, see the Electromigration Models section in the Liberate Details chapter.

This variable must be used before **char_library**.

Example:

```
# Set the EM Clock Frequency to 20MHz
set_var em_clock_freq  20e6
```

em_data_file

<**string**> The name of the Spectre EM technology file. Default: “”

Specify the Spectre APS EMIR data file needed to characterize electromigration. If the em_tech_file is also provided, the em_data_file is ignored.

For more information, see the Electromigration Models section in the Liberate Details chapter.

This variable must be used before **char_library**.

Example:

```
set_var em_data_file "/home/work/MyTech.dat"
```

em_period

<**value**> The EM clock period in seconds. Default: “”

Specify the clock period at which to characterize electromigration (EM). The value specified must be a number in seconds.

The `em_clock_freq` overrides `em_period`. It is recommended that only one of these two variables is specified. For more information, see the Electromigration Models section in the Liberate Details chapter.

This variable must be used before **char_library**.

Example:

```
# Set the EM Clock Period to 50nS
set_var em_clock_freq  50e-9
```

em_tech_file

<**string**> The name of the QRC technology file. Default: “”

Specify the QRC technology file to be used to characterize electromigration. It is recommended to use a QRC technology file specified using this variable instead of the Spectre APS EMIR data file specified using `em_data_file`.

For more information, see the Electromigration Models section in the Liberate Details chapter.

This variable must be used before **char_library**.

Example:

```
set_var em_tech_file "/home/work/MyTech.qrc"
```

em_user_string

<**string**> A string of Spectre EMIR configuration file commands. Default: “”

Specify a string containing Spectre-APS EMIR configuration commands that are included in the emir configuration file.

For more information, see the Electromigration Models section in the Liberate Details chapter.

This variable must be used before **char_library**.

Example:

```
set_var em_data_file "emirutil blech_length=longest_path"
```

enable_command_history

< 0 | 1 >	Set to ’1’ to enable command logging. Default 0 (do not log commands.)

extsim_ccs_option

<"options">	Options to be used for CCS timing characterization with external SPICE. Default: *set to extsim_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**. For accurate CCS current measurements it is advisable to use an accurate simulator setting (e.g. **"runlvl=6"** for Hspice). As delay and power characterization share the same simulation as CCS timing, this option will also impact 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 variable must be used before **char_library**.

extsim_cells_use_nodeset_for_io_pad

"list"	Options to be used for CCS timing characterization with external SPICE. Default: *set to extsim_option*

Use this variable to specify one or more cell(s) for which Liberate will use .nodeset instead of .ic in the simulation deck when the char_library -io option is specified. The .ic/.nodeset spice command is added on output pad nodes to help the external simulator find a DC solution. In some cases, the external simulator may find a circuit has difficulty reaching DC convergence with .ic but a .nodeset may help. In other cases, the opposite may be true. By default this parameter is empty, meaning there is no change in the Liberate default behavior. 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:

This variable must be used before **char_library**.

extsim_cmd

"string" Command string to be used to call the external SPICE simulator.
Default: s*ee below*

This option can be used to override the default command used by Liberate and call the external simulator. **IMPORTANT**: The simulator specified with **extsim_cmd** *must* agree with the **-extsim** option specified with char_library.

The default settings are:

For Hspice (Synopsys): extsim_cmd="hspice"

```
# call HSPICE
$extsim_cmd $extsim_cmd_option \
    -i $spicedir/sim.sp -o $spicedir/sim \
    >& /dev/null
```

For Spectre (Cadence): extsim_cmd="spectre"

```
# call Spectre
$extsim_cmd $extsim_cmd_option \
    sim.sp >& sim.lis
```

For Eldo (Mentor): extsim_cmd="eldo"

```
# call Eldo
$extsim_cmd $extsim_cmd_option \
    -o $spicedir/sim.lis -i $spicedir/sim.sp \
    >& /dev/null
```

For Finesim (Synopsys): extsim_cmd="finesim"

```
# call Finesim
$extsim_cmd $extsim_cmd_option \
    -spice sim.sp -o sim >& /dev/null
```

For XA (Synopsys): extsim_cmd="xa"

```
# call XA
$extsim_cmd $extsim_cmd_option \
    -hspice sim.sp -o sim >& sim.lis
```

This variable must be used before **char_library**. Example:

```
set_var extsim_cmd "spectre2"
set_var extsim_cmd_option "+log mylogfile"
```

File spectre2 contains:

```
#!/bin/sh
exec spectre $*
```

extsim_cmd_option

<"string"> Options to be passed to the extsim_cmd
Default: *see below*

This option can be used to override the default command options used by Liberate to call the external simulator.

We recommend that Spectre models are provided when using the Spectre simulator. If Hspice models will be used with Spectre, then this variable may need to include "+spice".

The default settings are:

- For HSPICE: extsim_cmd_option=""
- For Spectre: extsim_cmd_option="+lqt 0"
- For Eldo: extsim_cmd_option="-compat -nocou -nojwdb -queue -mgls_async"

To run Spectre in APS mode, set the **+aps** option:

```
set_var extsim_cmd_option +aps
```

This variable must be used before **char_library**.

extsim_constraint_option

<"options"> Options to be used for constraint characterization with external SPICE simulator.

This parameter specifies the list of options to be used by the external_simulator characterization for constraint (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 optionis is defined, the default for both the parameters is `extsim_option`.

This variable must be used before **char_library**.

extsim_deck_dir

<directory_name> Create a directory in which to store the output SPICE decks
Default: *follows $TMPDIR/decks.$PID*

This parameter specifies a directory to write all the SPICE decks created when using an external simulator. A tar'ed and gzipped file is written for each cell named <cellname>.tgz in the named directory. The directory must be visible to all server and client machines.

SPICE decks are only saved when the **char_library -extsim** option is enabled.

A file called **map.lst** will be 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. Sim 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 (e.g. from –vector of define_arc)
7. An optional CHK indicates this is an arc checking simulation

This variable must be used before **char_library**. Example:

```
# Set the directory for storing SPICE decks
set_var extsim_deck_dir "/home/char/decks"
char_library -extsim hspice -cells {INVX1 DFFX1}
```

extsim_deck_header

<"options"> — String of spice options written to external SPICE simulation deck header. Default: `.protect` for Hspice, `.protect` and `.option notrc NODCINFOTAB IMUDIV=0` for Eldo.

Use this control variable to overwrite what's written by default in the header of the extsim SPICE decks. By default HSPICE decks will have ".protect", and Eldo decks will have ".protect" and ".option notrc NODCINFOTAB IMUDIV=0"

These options might be superceded during simulation by duplicate options specified in `extsim_option`, or `extsim_leakage_option`, or similar options.

The **extsim_deck_header** variable is available so the external simulator commands can be provided directly to the external simulator without having Liberate process or review them. This variable 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.

This variable must be used before **char_library**.

extsim_deck_style

<merge | separate >	Request separate netlist and models from spice decks. Default: "merge" (keep netlist and models in-line)

Set this control variable to **separate** to request the separation of the netlist and the models from the extsim spice decks. The netlist and models will be saved into a separate file and loaded using the .include spice command.

This variable must be used before **char_library**.

extsim_exclusive

< 0 | 1 >	Control the SPICE engine used for pre-simulation. Default: 0

By default, Liberate uses Alspice for pre-simulation measurements. Set this variable to **1** to use the external simulation engine for all simulations when **char_library -extsim** is selected.

This variable is only intended for use in cases where Liberate cannot support the process model, netlists, or both (e.g. encrypted netlists/models, Verilog-A models, etc.). In these cases, **extsim_model_include** and **define_leafcell** should first be set properly. Note that the use of **define_leafcell** currently sets **extsim_exclusive**. In cases where the netlist cannot be processed, **extsim_flatten_netlist** may also need to be reset to 0.

Setting **extsim_exclusive** to **1** will degrade performance. It should only be used after trying all other options. The recommended and default value is **0**.

This variable must be set before **char_library**.

extsim_flatten_netlist

< -1 | 0 | 1 >	When set to 0, and **define_leafcell** is used, do not flatten the netlist in the extsim spice decks. Default: -1 (disabled)

This variable only applies when **extsim_model_include** is used. Set this variable to 0 to force Liberate to use **.inc** SPICE command to load cell netlist when **define_leafcell** command exists. By default (extsim_flatten_netlist = -1) Liberate 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.

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 variable must be set before **char_library**.

extsim_immunity_option

<"options">	Options to be used for characterization with external SPICE simulator. Default: *"" (use "runlvl=3 rmax=25" for Hspice, "" for Eldo or Spectre.)*

This parameter specifies the list of options to be used by external SPICE characterization for NIC (noise immunity curves). The **options** string is passed as a **.option** line in the SPICE deck. This option only works with Hspice. Default: "runlvl=3 rmax=25"

This variable must be used before **char_library**.

extsim_interactive

< 0 | 1 >	Set to enable external simulator interactive mode. Default: 0 (disabled)

Use this parameter to allow the external simulator, when it is enabled with **char_library -extsim**, to operate in an interactive mode. Running the external simulator in interactive mode will reduce 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: using Eldo in interactive mode can leave orphaned processes that will need to be manually killed if the Liberate/Variety run exits abnormally.

Note: This functionality only supports Eldo 2008.06 or later. Liberate uses the interactive mode, which only starts Eldo once upfront, instead of starting Eldo for each simulation deck. Users 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, then either correct the settings of the previously mentioned commands or disable **extsim_interactive** by setting it to **0**.

This variable must be used before **char_library**. Example:

```
set_var extsim_interactive 1
```

extsim_leakage_option

<"options">	Options to be used for leakage characterization with external SPICE simulation. Default: *set to extsim_option*

This parameter specifies the list of options to be used by external SPICE characterization for leakage characterization. The **options** string is passed as a **.option** line in the SPICE decks Liberate creates for leakage characterization.

This variable must be used before **char_library**.

extsim_lic_keep

< 0 | 1 > Enable the Hspice -cc mode. Default: 0 (disabled)

Use this parameter enable the Hspice -cc mode of operation. The external simulator must be enabled with **char_library -extsim**. The recommended value for this variable is **1** because it can significantly improve runtime when using *Hspice* as the external simulator. The version of Hspice being used must be 2008.09 or newer. If a version of Hspice is being used that does not support keeping a license checked out, then this variable should not be used.

This variable must be used before **char_library**. 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 variable 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. (For example, Hspice has a limit of 1024 chars.)

Setting this to 0 means no limit.

This variable must be used before **char_library**.

extsim_model_include

<"model file"> Specify full path to a file that will load the spice models. Default: use flattened models

Use this option to specify a full path to a file that will load the models when using an external spice simulation engine. An error will result if a full path is *not* provided. (See error messages below.)

Normally, Liberate will use flattened models in the external simulation input decks. When this variable is used, Liberate will use the file specified instead of the flattened models in the external simulation input deck. Liberate will place a statement in the extsim spice decks similar to this:

```
.include <extsim_model_include_file>
```

Here's an example of the extsim_model_include:

```
set_var extsim_model_include "/home/joe/models/include_ff"
    ...
char_library -extsim hspice
```

Where include_ff looks like:

```
.lib '/home/joe/models/models.l' ff
```

Error messages:

File does not exist:

Error : The extsim_model_include file <filename> cannot be found! Check the variable 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 variable 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.

This variable must be used before **char_library**.

extsim_model_include_leakage

<"model file">	Specify full path to model file. Used with HSPICE. 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 variable must be used before **char_library**. Example:

```
set_var extsim_model_include_leakage "/home/myDir/model.inc"
```

extsim_model_include_mode

< 0 | 1 | 2 >	Add 1e-5 Ohm resistor to all internal cell nodes. Default: 2 (Do not add resistor for leakage)

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.

0: Add 1e-5 Ohm resistor to all internal nodes of a cell. (Behavior of 2.4p1 and earlier releases.)

1: `.nodeset` and `.ic` statements in extsim spice decks for leakage will reference internal nodes directly instead of thru a 1e-5 Ohm resistor.

2: `.nodeset` and `.ic` statements in all extsim spice decks will reference internal nodes directly. (Default & Recommended)

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 will issue a warning and automatically set this variable to 2.

This variable must be used before **char_library.**

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.

Example:

```
# Set the directory for storing extsim monitor SPICE decks
set_var extsim_monitor_deck_dir "/home/char/decks_monitor"
```

See also:

extsim_monitor_enable
extsim_monitor_timeout

This variable must be used before **char_library**.

extsim_monitor_enable

<0 | 1>	Enable an external monitor script for an external simulator. Default: 0 (disabled)

This enables an external simulator monitoring script that can detect if the SPICE simulation is hanging, and will take corrective action. If necessary, Liberate will re-submit a simulation up to "`retry_count`" times before skipping the job. The only external simulator currently supported for monitoring is HSPICE.

Note: Only certain versions of HSPICE have been observed to hang indefinitely. Many of these hanging conditions have be resolved by updating the version of HSPICE or the process models.

The recommended setting is 0 (disabled). If it is necessary to enable this functionality to work around hanging conditions with an external simulator, we recommend you update the simulator version.

The actual script is "extsim_monitor.sh" in the $ALTOSHOME/bin directory, but we do not recommend that users modify this script.

See also:

extsim_monitor_deck_dir
extsim_monitor_timeout

This variable must be used before **char_library**.

extsim_monitor_timeout

<time> The time in seconds that an extsim process is inactive before action is taken by the extsim_monitor. Default: 1800 (30 minutes)

This variable 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 variable `exstim_monitor_enable` must be set in order for this variable 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.

Example:

```
# Set the extsim monitor timeout to 5 minutes
set_var extsim_monitor_timeout 300
```

See also:

extsim_monitor_enable
extsim_monitor_deck_dir

This variable must be used before **char_library**.

extsim_mpw_option

<"options"> 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 optionis is defined, the default for both the parameters is `extsim_option`.

This variable must be used before **char_library**.

extsim_node_name_prefix

<string> Specifies the generated node prefix string. Default: "`altos_`"

Use this variable 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 variable must be used before **char_library**.

extsim_option

<"options"> Options to be used for characterization with external SPICE Default: "runlvl=5" for Hspice, "save=none" for Spectre and "eps=1e-7" for ELDO

This parameter specifies the list of options to be used by external SPICE characterization for delay, power or timing constraint characterization. The **options** string is passed as a *.option* line in the external SPICE decks Liberate creates for characterization.

Note: The last extsim_option will *overwite* the previous setting.

This variable must be used before **char_library.** Examples:

```
# 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 variable to specify simulator options to be used during the presim stage. Example:

```
set_var extsim_option_presim "ITL1=300"
```

This variable must be used before **char_library**

extsim_reuse_ic

< 0 | 1 | 2 | 3 > Reuses DC solution for a group of transient simulations when Spectre is the external simulator. Default: 0 (Note: for Spectre-SKI this is fixed internally to 3)

This variable 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.)

0: Do not reuse DC solutions during `.alter` simulations. (Default)
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.)

This variable must be set before **char_library**.

extsim_sanitize_param_name

< 0 | 1 > Set this variable to enable sanitizing of node names in the output Spice deck. Default: 1

Enable 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: `\`, `<`, `>`, etc.)

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 variable must be set before **char_library**.

extsim_save_failed

<none | deck | all | log> Save SPICE decks for failing simulations. Default: all

This parameter is used to save SPICE decks and listings for failed simulations. If set to **deck,** then only the input SPICE deck is saved. If set to **all**, then both the input deck and the output listings are saved. If set to **log**, then only the input deck and output log file from the simulator are saved. The data is saved in the directory defined by the **extsim_deck_dir** variable.

Output SPICE decks are only saved when the **char_library -extsim** option is enabled.

This variable must be used before **char_library**.

extsim_save_passed

<none | deck | all | log> Save SPICE decks for passing simulations. Default: none

This parameter is used to save SPICE decks and listings for successful simulations. If set to **deck,** then only the input SPICE deck is saved. If set to **all,** then the input deck and the output listings are saved. If set to **log**, then only the input deck and output log file from the simulator are saved.

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** variable.

Output SPICE decks are only saved when the **char_library -extsim** option is enabled.

This variable must be used before **char_library**.

extsim_save_verify

< 0 | 1 | 2 > Generate additional SPICE decks to verify setup, hold, recovery and/or removal. Default: 0

When set to a **1**, this parameter is used to enable 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** variable and have a **"_sweep.sp"** suffix. Note that minimum pulse width SPICE decks will not be created.

When **extsim_save_verify** is set to **2**, **extsim_save_passed** will always be set 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. Note that this requires that all simulations will still be performed if **extsim_save_verify** is set to **2**.

Output SPICE decks are only saved when the **char_library -extsim** option is enabled.

This variable must be used before **char_library**. 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" Command used 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 variable must be used before **char_library**.

Example:

```
# Disable SPICE deck compression
set_var extsim_tar_cmd ""
```

extsim_timestep

<value> The time step to use for external SPICE simulation.
Default: 1e-12 seconds (1ps)

This parameter sets the time step for the external SPICE simulator.

This variable must be used before **char_library**. Example:

```
# Set the time step for external SPICE
set_var extsim_timestep 2e-12
```

extsim_tran_append

<"options"> Additional options to append to .tran. Default: "" (none)

This parameter is used to add extra options to the **.tran** statement.

This variable must be used before **char_library**. Example:

```
# Set conservative mode for Spectre
set_var extsim_tran_append "errpreset=conservative"
```

extsim_use_node_name

< 0 | 1 > Set this to 1 to request real node names in extsim spice decks
Default: 0 (map node names to numbers)

Set this control variable to a **1** to disable node to number mapping in the extsim spice decks. Default: 0 (map node names to numbers).

When **extsim_use_node_name** is set to a **1**, if a port name has an escape character "\", then Liberate will remove this character and replace it by '_' by default since spice simulators (hspice, eldo, spectre) do not support this character in voltage/current source commands.

This variable must be used before **char_library**.

floating_channel_bias

< value> Bias the initial conditions for floating channel nodes in Spice decks.
where, 0 <= value <= 1. Default is 0.5.

Use this variable to bias the initial conditions for floating channel nodes in Spice decks when enabled by setting the variable **floating_channel_mode** 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.

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.

This variable must be set before **char_library**.

floating_channel_mode

< 0 | 1 | 2 | 3> Enables various algorithms to initialize floating channel nodes. Default: 0

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 variable can be used to enable various algorithms to initialize floating channel nodes.

0: Initialize floating channel nodes if they drive the gate of a loading transistor. (Default and recommended)
1: Never initializes floating channel nodes and leaves it to the simulator to determine the initial condition.
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 variable **floating_channel_bias**.

This variable must be set before **char_library**.

force_avg_default_select_order

<0 | 1> Enforces a pre-determined order for selecting average default groups. Default: 0

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 variable is needed to resolve differences in default groups for cells with exactly two states.

0: Liberate will use its internal ordering to decide the default group. (Default)
1: Enforces a deterministic ordering. (Recommended.)

This variable must be set before **write_library**.

force_condition

< 0 | 1 | 2 | 3 > Controls whether to include conditional `when` and `sdf_cond` attributes for single or binate timing groups. Default: 1

0: Turns off 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: Output `when` for binate arcs (example: XOR) or constraint arcs; this is an aid to Verilog. (Default)

2: In addition to mode **1**, always output `when` (example: for AND gate.)

3: In addition to mode **1**, provide state dependent clock->output arcs even when they're non-symmetrical between rise and fall.

This variable must be used before **char_library**. Example:

```
# Turn off conditions on single timing groups
set_var force_condition 0
```

force_default_group

< 0 | 1 > Control whether to include a default group.
Default: 0 (default group may not always be included)

By default, Liberate will not always output a default group. Setting **force_default_group** to **1** will request Liberate to always output a default group, even if the group is redundant. This can occur when all states are exhaustively enumerated.

This parameter may be used after **char_library**. Example:

```
# Require output of default group
set_var force_default_group 1
```

force_edge_timing_type

< 0 | 1 > Model single-sided edge triggered arcs as rising_edge or falling_edge. Default: 0 (model as combinational arcs)

Set this variable to **1** to force Liberate to output a timing type of *rising_edge* or *falling_edge* for single sided edge triggered arcs (e.g. a clock to output arc in a clock gating cell). By default, Liberate will only output edge timing types (rising/falling_edge) when the timing group has both rising and falling data. If only one direction of data exists, the timing type will get converted to a timing type of combinational_rise/fall. This timing type change was implemented because early versions of Synopsys Library Compiler did not permit one-sided "edge" timing types, but the more recent versions support this. Default: **0** (convert single sided edge timing type to *combinational_rise* and *combinational_fall*. The recommended setting for this variable is **1** with 2009 and later versions of LC (Synopsys Library Compiler).

This variable must be set before **char_library**.

force_leakage_if_no_pg_pin

< 0 | 1 > Enable output of leakage when no related_pg_pin is available.
Default: 1

Set this variable 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. When set to 0, Liberate reverts back to pre-3.0 behavior where leakage groups will not be output if there is no related_pg_pin.

IMPORTANT: The **force_leakage_if_no_pg_pin** variable 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** variable. The default leakage power for that cell will not be output to the library, without regard for the value of the **keep_default_leakage** variable.

This variable must be set before **write_library**.

force_related_power_pin

< 0 | 1 | 2 > Force the output of related_power_pin groups for cells. Default: 1

If there is no related pg pin found in the netlist of a cell, the **force_related_power_pin** variable will control 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**. Setting this variable to 2, forces this behavior for every cell. The default value of 1 forces this behavior only for cells that have more than one pin (Antenna cells are not forced) Set this variable to 0 to not force the output of related_power_pin data. The recommended setting is 2.

This variable must be set before **write_library**.

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)

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 variable to force all specified supplies to be output in pg_pin groups.

0: pg_pin information for unconnected power supplies will *not* be output. (Default)
1: pg_pin information for unconnected power supplies will be output.

This variable must be set before **char_library**.

group_attribute

<"attribute_name"> Creates a cell group based on the named attribute. Default: none

A cell group may be created explicitly by the define_group command, or implicitly via cells having a common cell level attribute. This works with a **read_library / write_template** flow, and groups cells together that have the same value for a selected attribute (for example, "cell_footprint").

Example:

```
set_var group_attribute "cell_footprint"
```

This will group together all cells that have the same "cell_footprint" attribute. The attribute name will be used as the group name. Note: the default common attribute is "cell_footprint" but others may be used.

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 variable is used to specify a time limit to wait for a client process to communicate with the master Liberate 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 variable must be used before **char_library**. 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 variable 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 variable must be used before **char_library**. Example:

```
# Set the heartbeat timeout to 10 minutes
set_var heartbeat_timeout 600
```

hidden_power

< 0 | 1> Turn on 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 (i.e. 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 variable must be used before **char_library**. Example:

```
# Disable hidden power calculations
set_var hidden_power 0
```

ibis_compensate_odt

< 0 | 1 > Turns on compensation calculation for weak pullup/pulldown On-Die Termination (ODT) I/O cells. Default: 1 (on)

If **ibis_compensate_odt** is set to 1 (default), then Liberate will calculate the PAD pin current Ig at VSS, and current Ip at VDD (from total clamp IV data) and compare them to the parameter **ibis_odt_min_current** (Default: 2e-5; units in mA). If one of the two calculated/measured currents (Ip or Ig) is larger than **ibis_odt_min_current**, then Liberate will re-calculate the power clamp and GND clamp I-V table to remove (compensate) the ODT or weak pull-up/down current (which is already included in pullup/pulldown I-V table.)

Note: see also ibis_odt_min_current

This variable must be used before **char_library**.

ibis_has_weak_hold

< 0 | 1 > Specify to request .NODESET instead of .IC in extsim decks for output pin initial condition. Default: 1

When preparing IBIS simulations, Liberate will use the .IC spice command to initialize Hi-Z pins in the spice deck. If set to 1, in IBIS mode (see **char_library -ibis**) Liberate will use .NODESET instead of .IC in the spice deck.

The initial voltage on a node will be set to a supply voltage. When the output does not swing full rail, this voltage might be incorrect which can cause the spice engine difficulty in finding a DC solution. It is advisable to set this variable when the spice engine complains or takes too long finding a DC solution and the user knows the holding strength on the output pin is weak (no full vdd/vss swing).

This variable must be used before **char_library**.

ibis_iv_max_step_factor

<value> Specify the maximum voltage step in the filtering algorithm as a factor of ibis_iv_step. Default: 1.5

This variable is used to control the maximum voltage step in the filtering algorithm when writing out I-V tables. IBIS contains up to 3 process corners each with a set of waveforms (I-V table) that have to be combined into a data table with the same rows (the V for I-V table). Also, the total number of rows are limited by different IBIS versions (for example: 100 voltage rows/steps for the I-V table. Liberate needs to reduce up to 1000 rows into 100 rows).

Liberate uses a proprietary algorithm to retain the best waveform points globally. By default, Liberate will use 1.5 * 0.05 = 0.075 mV (ibis_iv_max_step_factor * ibis_iv_step) to guarantee there should be at least one data point for every such voltage range.

This variable must be used before **char_library**.

ibis_iv_mode

< 0 | 1 > Enables differential-pin IBIS characterization. Default is 0 (Preserve previous behavior.)

Set to 1 to enable input differential-pin IBIS characterization; not needed for non-differential input pin model. Default: 0 (off).

This variable must be used before **char_library**.

ibis_iv_step

<value> The IBIS voltage step. Default: 0.05v

Sets the IV voltage step used in the IBIS file when characterizing I-V data such as power_clamp, gnd_clamp and pull_up, pull_down table. Default is 0.05V.

This variable must be used before **char_library**. Example:

```
# Set the acceptable glitch output peak to 10%
set_var ibis_iv_step 0.1
```

ibis_odt_min_current

<value> User-defined threshold for IBIS weak pullup/pulldown compensation calculation. Default: 2e-5 mA (20 nA)

See **ibis_compensate_odt** *for full explanation.*

This variable must be used before **char_library**.

ibis_sim_duration

<value> The IBIS simulation duration. Default: 20e-9

Set the simulation duration for the IBIS related simulations. Use the ibis_sim_duration and ibis_tend_factor (default 1.8) to make the simulation end time consistent in all 3 IBIS corners (typ/min/max).

This variable must be used before **char_library**. Example:

```
# Set the acceptable glitch output peak to 10%
set_var ibis_sim_duration 30e-9
```

ibis_t2b_cmd

"string" Command string to be used to change the t2b executable. Default: t2b.

This option can be used to override the default command used by Liberate and call the external executable. For example, if user wrote a wrapper script for t2b with the path name of `/bin/t2bwrapper.exe`, then setting this parameter to `set_var ibis_t2b_cmd "/bin/t2bwrapper.exe"` overwrites the default setting.

ibis_tend_factor

<value> The IBIS transient simulation factor. Default: 1.8

Set the IBIS transient simulation end time which is to be multiplied by the original Liberate simulation end time estimation. The final end time will be compared to the end time specified by **ibis_sim_duration** (if greater than 0) and take the larger value to use for the simulation end time.

The **ibis_tend_factor** and **ibis_sim_duration** variables exist to make sure Liberate can see the voltage/time waveform tail. The default Liberate simulation end time estimation might not be long enough for some analog IO pad cells.

This variable must be used before **char_library**.

ibis_vt_max_num_pts

<value> Specifies the maximum number of V-t time points. Default: 1000.

Set this variable to the maximum number of time points that will be output in a V-t table in an IBIS model. The value must be >= 40. If the simulation output has fewer data points than the value set by this variable, then all of the available points will be printed. If the simulation output has more points than the value set by this variable, the data will be reduced to have the specified number of points. The value specified should be greater than or equal to the `ibis_vt_min_num_pts` value.

Example:

```
set_var ibis_vt_max_num_pts 500
set cell "BC335"
set file "bc3350.ibs"
set ldbs { ibis_BC335_typ.ldb.gz ibis_BC335_min.ldb.gz ibis_BC335_max.ldb.gz }
write_ibis_file $file $ldbs $cell
```

The generated rising/falling V-t waveform will have up to 500 time points written in the IBIS file as long as the original V-t data has more than 2 time points stored in the ldb.

This variable should be set prior to the **write_ibis_file** command.

ibis_vt_min_num_pts

< value> Specifies the minimum number of V-t time points. Default: 40

Set this variable to the minimum number of time points that will be output in a V-t table in an IBIS model. The value must be >= 2. If the simulation output has fewer data points than the value set by this variable, then Liberate will use linear interpolation to add data points at a sparse area of the waveform curve. The value specified should be less than or equal to the `ibis_vt_max_num_pts` value.

Example:

```
set_var ibis_vt_min_num_pts 500
set cell "BC335"
set file "bc3350.ibs"
set ldbs { ibis_BC335_typ.ldb.gz ibis_BC335_min.ldb.gz ibis_BC335_max.ldb.gz }
write_ibis_file $file $ldbs $cell
```

The generated rising/falling V-t waveform will have at least 500 time points written in the IBIS file even if there are fewer V-t time points stored in the ldb.

This variable should be set prior to the **write_ibis_file** command.

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 variable must be used before **char_library**. 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 variable must be used before **char_library**. 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 prediction of the circuit behavior, then `init_clock_period` is silently disabled for this arc.

2. Use the behavior as in #**1** above, but apply the `input_constraint_period` only to the related_pin. No side pins will be pulsed during the initialization sequence. (Recommended)

This variable must by set before **char_library**.

init_comb_num_cycles

<integer>	Number of times the related_pin will be toggled. Default: 1

This variable 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** variable is set to a positive number will **init_comb_num_cycles** be used. Otherwise, the related_pin will not be toggled.

This variable must be used before **char_library**. 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 variable 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 variable while the number of toggles is set by the **init_comb_num_cycles** variable.

This variable must be used before **char_library**. Example:

```
set_var init_comb_related_pin_period 10e-9
```

init_constraint_period

< value >	Clock period (in seconds) used for circuit initialization during constraint 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 the circuit before constraint circuit simulation. Default: no initial clock pulse is used.

This variable must by set before **char_library**. 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 variable must be used before **char_library**.

init_constraint_period_check_mode

< 0 | 1 >	Controls how vector pre-pulsing is applied. Default: 0

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.

0: Check to make sure the entire vector is identical before-and-after pre-pulsing. (Default)
1: Loosens the check to just the probe node.

This variable must be used before **char_library**.

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 variable must be used before **char_library**. 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 variable must be used before **char_library**.

init_pin_hidden_num_cycles

<integer>	Number of times the related_pin will be toggled. Default: 1

Use this variable 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` variable is used only when the `init_pin_hidden_period` variable is set to a positive number. Otherwise, the related_pin will not be toggled.

This variable must be used before **char_library**.

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)

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.

0: Standard method. (Default)
1: Extra effort is used to make certain the cell is properly initialized. (Recommended)

This variable must be used before **char_library**.

input_noise

<dc | hyper | both> SI constructs to write for input pins. Default: hyper

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. The **dc** option will cause **input_voltage** DC noise level attributes to be written. The **hyper** option outputs noise rejection curves (**hyperbolic_noise_low, hyperbolic_noise_high**). The **both** option enables both DC noise and hyperbolic constructs to be written.

This parameter may be used after **char_library**. 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 (Default)

1: Create `input_voltage`, `output_voltage` groups and use VDD/GND (rail voltages) for vih,voh and vil,vol:
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:
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
  }
```

keep_dcap_leakage

< 0 | 1 > Controls whether to keep the default DCAP leakage group
Default: 0 (Don't keep.)

Set this parameter to ’1’ to keep the default leakage groups for decap cells. (See also, keep_default_leakage_group.)

This variable can be used after **char_library**.

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.
1: *(Reserved)*
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 are as follows:

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 variable can be used after **char_library**.

keep_user_defined_arc_failed_data

< 0 | 1 > Keep user specified arcs (see **define_arc**) even if they cannot be characterized.
Default: 1 (Don't 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 variable 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 variable must be used before **char_library**.

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 variable must be used before **char_library**. 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 variable 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"
```

Please note these 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 variable must be used before **char_library.**

ldb_save_all_cells

< 0 | 1>	Controls if cells in the input ldb not being re-characterized are written out into the output ldb. Default: 1

When set to 1, the **ldb_save_all_cells** variable ensures that cells in the input ldb that aren't being re-characterized are still written out into the output ldb. To get the pre 3.0p2 and prior behavior, set this variable to 0. The recommended setting is 1, which is also the default.

This variable must be used before **char_library.**

leakage_accuracy_mode

< 0 | 1>	Improves leakage accuracy. Default: 0

When set to 1, leakage accuracy is improved. (Default: 0) The recommended setting is 1.

This variable must be used before **char_library.**

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 variable must be used before **char_library**.

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 variable to 0 will stop Liberate from adding the missing leakage groups, thus avoiding an LC error. (Default: 1)

This variable must be used before **char_library**.

leakage_cell_attribute

< 0 | 1 > Output cell_leakage_power. Default: 1

Set this variable to **0** to omit *cell_leakage_power* attributes from the output library.

This variable can be used after **char_library**.

leakage_float_internal_supply

< 0 | 1 | 2 > Affects leakage measurements for power switch cells. Default: 1

Set this variable 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, then leave the internal_supply pin floating.
- If the power switch is off, then 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).

The recommended setting is 0.

1: The internal_supply pin will always be floating when measuring leakage.

2: The behavior will match the setting of 1 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.

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
```

This variable must be used before **char_library**.

leakage_force_tristate_pin

< 0 | 1 > Set to force tristate pins output *cell_leakage_power* (Default: 0)

Set this variable to **1** to force a tri-stated pin to ground during leakage simulations. This variable can be used to help match legacy libraries. When set to **0** (default), Liberate will allow tristate outputs to float. This variable must be used before **char_library**.

leakage_merge_state

< 0 | 1 | 2 > Specify the leakage merge algorithm. (Default: 0)

Use this variable to tell Liberate how to select the leakage value to report when multiple measurements are being merged. Merging of leakage states might be needed, for example, when the there are user defined leakage states that do not include all pins. The recommended value is **0** (default). Use with **toggle_leakage_state** set to **1** to drop output variables from the leakage *when* string, then worst (or best) case the leakage value and remove the **leakage_power** groups with the same *when* strings. Supported values are **0** (default), **1** (min) or **2** (max). This functionality can be used to tune the reported leakage to existing libraries. This variable must be set before **char_library**.

leakage_mode

< 0 | 1 | 2 > Set the leakage computation mode (Default: 1 - multiply the leakage current by the supply voltage)

This option will specify the method used when computing the DC leakage to report in the .lib. 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	$(-I_{VDD} \times V_{SWING}) - (I_{INPUT\_HIGH} \times V_{INPUT\_SWING})$
	VSS	$(I_{VSS} \times V_{SWING}) + (I_{INPUT\_LOW} \times V_{INPUT\_SWING})$
1	VDD	$(-I_{VDD} \times V_{VDD}) - (I_{INPUT\_HIGH} \times V_{INPUT\_HIGH})$
	VSS	$(I_{VSS} \times V_{VSS}) + (I_{INPUT\_LOW} \times V_{INPUT\_LOW})$

Note: All current in the formulae above are the actual measured currents. In the case where multiple supply rails (like VDD, VDDL) are associated with a single ground rail (VSS), then 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 formulae assume that leakage_add_input_pin is set to **1**. When this variable is set to **0,** then all input leakage components in above formula are ignored.

This variable must be used before **char_library.**

leakage_model_internal_pin

< 0 | 1 > Controls if internal pins are used in leakage "when" state. (Default: 1)

When this variable is set to 0, Liberate will remove internal pins from being modeled. The `internal_pin` group and the `internal_pin` in the leakage when condition will not be output.

This variable can be used before **char_library.**

leakage_precision

<value> Specify leakage precision (Default: "%g")

Set this variable 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 variable can be used after **char_library**.

leakage_ramp_vsrc

< 0 | 1 > Initializes voltage sources to 0V at time zero and ramps to its intended value at the time equal to ½ of **leakage_sim_duration**. (Default: 0)

If this variable is set to 1, Liberate initializes all voltage sources to 0V at time zero, then ramps those voltage sources to its intended value at a time equal to ½ of the **leakage_sim_duration** variable. If **leakage_sim_duration** is not set or if it's set to a value of less than 1ns, then the **leakage_sim_duration** variable will be automatically forced to 1ns. The **leakage_ramp_vsrc** variable is particularly helpful with leakage simulation where HSPICE errors out due to its inability to find a DC solution.

This variable should be used before **char_library**.

leakage_sim_duration

<value> If non-zero, use transient simulation after the specified delay for leakage. Default: 0 seconds (use dc solution)

Use this variable 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 variable works with Alspice and with external simulators.

This variable 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 variable is not set, then the power supply ramping for leakage simulations will follow the sim_init_duration.

This variable must be used before **char_library**.

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.

This variable must be used before **char_library**. 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 variable must be used before **char_library**. Example:

```
# Set the revision to 2.0
set_var library_revision "2.0"
```

library_revision_mode

< 0 |1 > Affects value of **library_revision** text string. Default 1.

If library_revision is specified as "1.0", then:

0: Inserts library_revision in this form: revision : "$Revision: 1.0 $";
1: Inserts library_revision in this form: revision : "1.0"; (Default)

This variable must be used before **char_library**.

lic_max_timeout

<value> Specifies the duration of time, in seconds, to wait for the required licenses to be acquired. Default: 600 (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 variable **lic_max_timeout** 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 shell environment variable ALTOS_LIC_MAX_TIMEOUT will override the value set by this variable in the Tcl file. For more information, see ALTOS_LIC_MAX_TIMEOUT.

This variable must be used before **char_library**.

lic_queue_timeout

<value>	Specifies the duration of time, in seconds, to wait for the required licenses to be acquired. Default: 60 (seconds)

The shell environment variable ALTOS_LIC_CHECK_ALT_TIMEOUT will override the value set by this variable in the Tcl file. For more information, see ALTOS_LIC_CHECK_ALT_TIMEOUT.

This variable must be used before **char_library**.

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 variable must be used before **char_library**.

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 variable must be used before **char_library**.

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 **char_library**. Example:

```
# Set the logic AND, OR and NOT string to &&, ~, |
set_var logic_and "&&"
set_var logic_not "~"
set_var logic_or "|"
```

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 variable must be used before **char_library**. 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: 0

This variable issues a warning when read_ldb reads in an ldb that contains the **altos_error_flag** set to true. If set to 1, all children groups of the group containing the failed data have their values set to DBL_MAX/3 (inf in .lib).

This variable is used after **read_ldb** and before **write_library**.

mark_failed_data_replacement

<value> Replacement value for failed data
Default: 5.99E+307 (large library unit)

When **mark_failed_data** is enabled, this variable specifies the value to be plugged into the library for the failed data.

This variable is used before **write_library**.

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 may be used after **char_library**. 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: 0

0: Liberate will select the max of the maximum index_2 values. (Default)
1: Liberate will select the min of the maximum index_2 (load_index) values.

The recommended value is 1.

This variable must be used before **char_library.**

Example:

```
set_var max_capacitance_attr_mode 1
```

max_capacitance_auto_mode

< 0 | 1 > Select the computation mode for tiehi/tielo cells. Default: 0

When this variable is set to **1**, the **char_library and write_vdb** arguments **-auto_index** or **-auto_max_capacitance** will calculate the *max_capacitance* pin attribute for TieHi and TieLo cells by dividing the *max_transition* by the Rds. To measure the Rds, connect a 100kOhm resistor to the opposite supply and compute the Rds (Resistance drain to source) as follows:

TieHi_Ion = (Vdd -Vth)/Rds = Vth/Rl

TieLo_Ion = (Vdd -Vtl)/Rl = Vtl/Rds

Max_cap = max_transition / Rds

Where: Vth = The TieHi steady-state output pin voltage.
Vtl = The TieLo steady-state output pin voltage.

This parameter must be used before **char_library**.

max_capacitance_derive_limit_maxload

< 0 | 1 > Limits the max_capacitance attribute to the largest index_2 characterized load value. Default: 0

1: Limits the max_capacitance attribute to the largest index_2 characterized load value when write_library -derive_max_capacitance is enabled. (Recommended)
0: Allows max_capacitance to be larger than the largest index_2 characterized value. (Default)

This variable must be specified before **write_library**.

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 indices. Default: 1.

This parameter may be used after **char_library**. 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 variable. Default: 1 (Farads).

This variable must be used before **char_library**. Example:

```
# Set maximum allowed capacitance load
set_var max_capacitance_limit 1e-9
```

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 variable must be used before **char_library**. 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 indices)

This parameter is used to enable automatic creation of the indices for **si_immunity** tables. This parameter sets the maximum allowable noise-glitch width. Setting this parameter overrides indices defined in noise immunity templates. This parameter should always be set when characterizing noise immunity.

This variable must be used before **char_library**. 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 variable must be used before **char_library**. 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 (Seconds)*)

Use this parameter to set a maximum value allowed for all *max_transition* attributes. This limit is applied after max_transition_factor.

This parameter may be used after **char_library**. Example:

```
# Set max transition attribute limit to 1ns
set_var max_transition_attr_limit 1e-9
```

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 indices.

This parameter may be used after **char_library**. Example:

```
# Set max transition factor
set_var max_transition_factor 0.66
```

max_transition_for_outputs

<-1 | value | max>	Requests **max_transition** on output pins. Default: -1 (output pins do not have the attribute max_transition)

Use this variable to request the **max_transition** attribute on output pins. The default is to not output this attribute on output pins. Set this variable to a **value** to force a specific max_transition value on an output pin or to max to use the maximum value of all the rise/fall_transition values from all the arcs ending at that pin.

This variable must be used before **write_library**.

max_transition_include_power

< 0 | 1 >	Includes power arcs when determining **max_transition**. Default: 0 (max_transition is determined from timing arcs only.)

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. Set this variable to **1** to tell Liberate to include power arcs in the search for determining **max_transition**.

This variable must be used before **write_library**.

measure_cap_lower_fall

<value>	The % point on the cell input waveform to use for measuring falling input capacitance to. Default: 0.5 (50% of supply)

measure_cap_lower_rise

<value> The % point on the cell input waveform to use for measuring rising input capacitance from. Default: 0.0 (0% of supply)

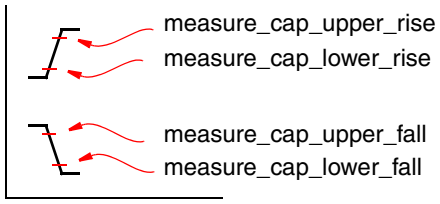
measure_cap_upper_fall

<value> The % point on the cell input waveform to use for measuring falling input capacitance from. Default: 1.0 (100% of supply)

measure_cap_upper_rise

<value> The % point on the cell input waveform to use for measuring rising input capacitance to. Default: 0.5 (50% of supply)

The above parameters are used to control how input capacitance is measured after SPICE simulation.



These variables must be used before **char_library**. 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
```

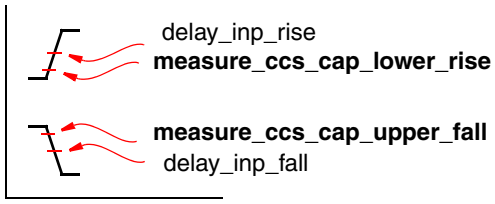
measure_ccs_cap_lower_rise

<value> The % 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 % point on the cell input waveform to use for measuring falling input CCS capacitance from. Default: 1.0 (100% of supply)

These variable are used to measure the first portion (C1) of the CCS receiver capacitance model; used in combination with the delay rise/fall thresholds:



These variables must be used before **char_library**. Examples:

```
# Set the CCS capacitance measurements
set_var measure_ccs_cap_upper_fall 0.3
set_var measure_ccs_cap_lower_rise 0.7
```

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 variable 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 which means the outputs are assumed to swing full rail. This variable only applies when the **io** argument to **char_library** is used.

This variable must be used before **char_library**.

measure_output_range_abstol

< minimum range Volts> Define the minimum output voltage swing range. Default: `0.100` (Volts)

This variable is used to specify the minimum supported output voltage swing range. It is enabled when the variable `measure_output_range=1`. 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 variable must be set before **char_library**.

measure_slew_lower_fall

<value> The % point on the cell output waveform to measure falling output transition times to. Default: 0.3 (30%)

measure_slew_lower_rise

<value> The % point on the cell output waveform to measure rising output transition times from. Default: 0.3 (30%)

measure_slew_upper_fall

<value> The % point on the cell output waveform to measure falling output transition times from. Default: 0.7 (70%)

measure_slew_upper_rise

<value> The % point on the cell output waveform to measure rising output transition times to. Default: 0.7 (70%)

The above parameters are used to control how output transition times are measured after SPICE simulation.

These variables must be used before **char_library**. 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 variable must be used before **char_library**.

mega_enable

< 2 | 3 > Default: 3

Use this variable to enable the use of an advanced algorithm refered to as "mega" mode. This mode improves inside_view preprocessing of large cells.

2: Enables mega mode for all cells in the library. It is not necessary to use the **define_cell** `-type mega` option.

3: (Default) 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 variable must be set before **char_library**.

merge_related_preset_clear

< 0 | 1 | 2 > Enables the merging of related preset and clear states. Default: 0

The arc which is caused by the de-assertion of a preset/clear signal may be considered a *combinational_fall/rise*, or a *clear/preset* type. When **merge_related_preset_clear** is set to a 1, the rise and fall tables from an asynchronous preset/clear related pin will be merged into a single timing group with the clear/preset type. When **merge_related_preset_clear** is set to a 2, both the clear and preset timing_type groups will be output.

By default, the preset(clear) will generate a rise(fall) transition, and the timing arc from the de-assertion of the preset/clear will be omitted. If this parameter is **0** (default), and combinational_risefall is set to **1**, then both the preset/clear arc and their associated de-assertion arc will be output in separate timing groups.

This variable can be used after **char_library**.

Example:

```
# Allow the merging of combinational timing types with
# preset/clear timing types.
set_var combinational_risefall 1
set_var merge_related_preset_clear 1
```

min_capacitance_for_outputs

< 0 | 1 > Set this variable to enable min_capacitance attribute for output pins. Default: 0

Set this variable 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 variable must be set before **char_library**.

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 to **char_library**. If not specified the minimum input pin capacitance found in the library is used.

This variable must be used before **char_library**. 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. Default: 0

The `minimum_period` will be calculated as 2 * max (mpw_high, mpw_low).

0: No minimum period calculation. (Default)
1: Calculate minimum period for clock (flip-flops) and enable (latch) pins only.
2: Calculate minimum period for clock/enable pins plus async pins.

This variable must be used before **write_library**.

min_period_when

"string"	Specifies the logic *when* condition for the `min_period` constructs. Default: ""

Setting this variable 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 variable works only in conjunction with the **min_period** variable.

This variable must be used before **write_library**.

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 variable must be used before **char_library**.

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.)

Example:

```
# Set min transition attribute limit
set_var min_transition_attr_limit 1e-12
```

This parameter may be used after **char_library**.

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 indices. Default: 1.

Example:

```
# Set min transition factor
set_var min_transition_factor 0.66
```

This parameter may be used after **char_library**.

min_transition_for_outputs

< -1 | "min" | value >	Requests a `min_transition` attribute on output pins. Default: -1 (Do not add this attribute.)

Use this variable to add the `min_transition` attribute on output pins.

-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 variable must be used before **write_library**.

min_transition_include_power

< 0 | 1 > Includes power arcs when determining `min_transition`. Default: 0 (`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. Set this variable to 1 to tell Liberate to include power arcs in the search for determining `min_transition`.

This variable must be used before **write_library**.

mpw_criteria

<delay | glitch> Specifies mpw failure criteria as delay or glitch. Default: glitch

This parameter can be used to specify the failure criteria to be used when measuring the minimum pulse width. Legal values are: **delay** and **glitch**. When delay is specified, the **constraint_delay_degrade** will specify the delay degrade value. When glitch is specified, the **mpw_glitch_peak** will specify the glitch height.

This variable must be used before **char_library**.

mpw_delay_use_active_edge

< 0 | 1 > Specify which clock edge to use when characterizing mpw. Default: 0

Set this variable 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 recommended setting for this variable is **1**.

This variable must be used before **char_library**.

mpw_glitch_peak

<ratio> Specify the MPW glitch peak failure threshold. Default: 0.95

To measure MPW (minimum pulse width) Liberate will apply a narrowing pulse to the input pin until the output pin fails. The default failure metric is to use a glitch peak. The glitch peak threshold is set by this variable. This variable will have not affect if delay has been selected as the failure criteria (see **mpw_criteria**).

This variable must be used before **char_library**.

mpw_input_threshold

<value> Defines the input threshold. Default: 0.99 (99% of Vdd)

This variable specifies the minimum height allowed when the clock pulse becomes triangular. Default is 99% of Vdd. The *min_pulse_width* constraint will be deemed to be violated if this occurs before.

This variable must be used before **char_library**.

mpw_linear_waveform

<value> Defines the input threshold. Default: 0

Set this parameter to request that Liberate use a linear waveform on the input pin when characterizing the minimum pulse width. By default, the same input waveform is used for mpw as is used for delay.

This variable must be used before **char_library**.

mpw_search_bound

<value> Defines the MPW initial search bound. Default: 5e-9

Use this control variable to control the initial search bound for mpw characterization. Default= 5e-9.

This variable must be used before **char_library**.

mpw_search_mode

< 0 | 1 > Enables fast algorithm for determining MPW. Default: 0

0: Default algorithm using bisection techniques. (Default)
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.) (Recommended)

This variable must be used before **char_library**.

mpw_skew_factor

<value> Defines the ratio of the input fall slew over the input rise slew. Default: 1.0 (rise = fall)

This variable enables skewing the ratio of the input rise versus input fall slew for calculating *min_pulse_width* constraints. The default is that the rise and fall input slews are identical.

This variable must be used before **char_library**.

mpw_slew

<value | min | mid | max> Defines the input slew for minimum pulse width characterization. Default: min

This variable 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 indices of the first delay arc where this clock signal is a related pin (e.g. a clock to Q delay arc on a flip-flop).

This variable must be used before **char_library**. 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 variable 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 variable is set, the slew applied to clock nets will be: **mpw_slew_clock_factor * mpw_slew**.

This variable must be used before **char_library**. 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 > Enables generating a one dimensional table of mpw values.
Default: 0 (generate mpw attributes only)

Use this parameter to enable the generation of a minimum pulse-width data table. When this parameter is set the mpw table will use the *index_1* table as specified by the **constraint_template** associated with the **define_cell** command for the cell. Characterizing mpw tables will increase runtime. By default, Liberate will only output mpw attributes.

This variable must be used before **char_library**.

mpw_vector_bin_mode

< 0 | 1 | 2 | 3 > Controls pre-check index point for binning vectors.
Default: 0 (no binning)

Set this 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 variable specifies which index point in the table should be used to determine the binning of vectors. Legal values are:

0: No binning (Default)
1: First index point
2: Middle index point
3: Last index point

This variable should be used before **char_library**. 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 variable must be used before **char_library**.

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. (Default)
1: Display a message when an attribute named in the **user_data_override** variable is overridden by data in the **write_library -user_data** file.

This variable must be used before **write_library**.

msg_limit_per_type_per_cell

<integer>	Controls the message limit for each message type. Default: `5`

Set this variable to output the maximum number of messages for each message type. The default value is `5`.

This variable should be set before **char_library**.

net_batch_mode

< 0 | 1 >	Sets Netbatch as the queuing system. Default: 0 (Do not use Netbatch.)

Allows user to specify Netbatch as a queuing system. Cannot be used together with `qsub_no_shell_mode`.

non_seq_copy_src_pin

<pin_name>	Name of pin for copy/transpose operation

non_seq_copy_dst_pin

<pin_name>	Name of pin for copy/transpose operation

These two variables 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 variable must be used before **write_library**. Example:

```
set_var non_seq_copy_src_pin CDN
set_var non_seq_copy_dst_pin SDN
```

non_seq_pin_swap

< 0 | 1 > Controls swapping of the constrained pin and related_pin when characterizing **nonseq_setup/hold.** Default: 1

0: Liberate will swap the pin and the related_pin.
1: Liberate will *not* swap the pin and related_pin. (Default)

This variable must be used before **char_library**.

nonseq_as_recrem

< 0 | 1 > The parameter converts arcs with timing-type *nonseq_setup/hold* into recovery/removal. Default: 0

When this variable is set, all arcs with the timing-type of *nonseq_setup/hold* arcs are converted into recovery/removal. By default, these arcs are not converted.

This variable can be used after **char_library**.

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 > The parameter requests the internal probe pins to be included in the .lib file. Default: 0 (Do not include internal pins.)

When this variable is set, all internal probe pins will be output in the .lib file with the type *internal_pin*. By default, these pins are not output into the .lib file.

This variable must be used before **char_library**.

packet_arc_licensing_mode

< default | reduce_clients | wait_for_clients > Controls license-handling for arc-based packet mode. Default: Wait for licenses on packet clients.

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.

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)
reduce_clients: The Liberate server performs block check-outs of Liberate_client and Spectre_char_opt licenses based on the availability at start. If insufficient licenses are available, then the number of clients is reduced to what is currently available.
wait_for_clients: Similar to "reduce_clients", but the server continues to query the license server for additional licenses. (Recommended)

There are two advantages of using either `reduce_clients` or `wait_for_clients`. With both these settings, the server is able to manage the license checkout. This results first in block check out of client licenses thereby, reducing the number of accesses to the Cadence license daemon which can prevent hitting a simultaneous access limit on extremely large installations. Second, the client licenses are held throughout the entire characterization. This maximizes the efficiency of the server license, which is usually the limiting factor.

This variable must be used before **char_library**.

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 variable 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 (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 variable must be used before **char_library**.

packet_client_resubmit_count

<number>	Specifies the number of times a failed LSF job should be resubmitted. Default: 0

This variable 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 variable must be used before **char_library**.

packet_client_timeout

<value>	Set a timeout value in seconds for client machines on the network. Default: 86400 (1 day in seconds)

This variable specifies a timeout limit (in seconds) for client machines on the network. If a packet client log file has not been updated for more than the number of seconds specified, Liberate will assume that packet has died. (This can occur because of a machine crash, or a signal such as "kill -9" that can't be trapped.) If a packet client is determined to be “dead”, then the server will not wait and move on to the next client.

This variable must be used before **char_library**.

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 to **0**.

This variable must be used before **char_library**.

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: cell

Liberate can distribute Parallel Packets in cell-based mode or arc-based mode.

cell: Cell-based mode. (Default and recommended)
arc: Arc-based mode

This variable must be used before **char_library**.

packet_rdb_mode

< 0 | 1 > Enable RDB in parallel packets. Default: 0 (enabled).

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, re-characterizations will take much less time, as Liberate can skip a simulation if the results are already present.

0: Disables RDB in parallel packets.
1: Enables RDB in parallel packets. (Default)

This parameter will be automatically set to 1 when `packet_mode=arc`. The recommended setting is `0` in cell packet flow.

This variable must be used before **char_library**.

parenthesize_not

< 0 | 1 > This puts parentheses around variable names that are negated using the exclamation mark (!). Default: 1

Liberate treats the *not* expression as represented with parentheses around the variable name in logic functions. Example: !(A)

0: Parentheses will not be added, i.e.: !A
1: Put parentheses around variable, i.e.: !(A) (Default)

This variable must be used before **char_library**. Example:

```
# Remove () from variable names
set_var parenthesize_not 0
```

parenthesize_sdf_cond

< 0 | 1 > Puts parentheses around all sdf_cond statement in output library. Default: 0

0: Do not put parenthesis around sdf_cond statements. (Default)
1: Put parentheses around sdf_cond statements.

This variable can be used after **char_library**.

parse_space_bang_is_comment

< 0 | 1 > Specifies whether to treat bang (!) preceded by a space as a comment. Default: 0

When set to the default of 0, this variable parses the netlist and does not treat bang (!) preceded by a space as a comment. Setting it to 1 accepts bang preceded by a space as a comment when parsing the netlist. This may help when parsing Eldo format netlists and models.

This variable must be used before **read_spice**.

pin_based_leakage

< 0 | 1 > The parameter disables pin-based leakage reporting Default: 1

Use this variable to control whether or not to output pin based **leakage_power** groups. Supported values are:

0: Output only one leakage_power per state condition.
1: Output leakage_power groups based on state and power.

This variable must be used before **char_library**.

pin_based_power

< 0 | 1 | 2 > The parameter enables a pin based power calculation. Default: 1

The above parameter is used to control how pin based power is characterized. When **pin_based_power** is enabled in mode **1**, Liberate will monitor power based on all the pins of a cell. If disabled (mode **0**) only the power in the power supplies (Vdd's) is monitored. Mode **2** is similar to mode **1** with the addition that Liberate will monitor the Miller capacitance current to each cell input. If the input is set to a positive/negative voltage, that current is added to the

vdd/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 was, it may not be switching in the correct way to trigger the Miller capacitance currents. See Liberate Details for more details on power characterization.

The parameter **pin_based_power 0** can co-exist with **voltage_map 1**. When this specific condition is encountered, the positive supply power will have CV^2 subtracted. In normal mode, both positive and negative supplies 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**.

This variable must be used before **char_library**. Example:

```
# Set the power calculation to monitor only VDD power
set_var pin_based_power 0
```

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 characterization "when" conditions and the specified condition while considering capacitances. Setting it to 0 will have it consider overlapping "when" states as well. Default is 0.

This variable must be used before **write_library**.

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 variable 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 variable to "" (the empty string) to request that if pin sorting is enabled (see **sort_pins**) then all pins are sorted alphanumerically.

Example:

```
set_var sort_pins 1
set_var pin_type_order {internals inouts outputs inputs}
```

This variable must be set before **write_library**.

pin_vdd_supply_style

< 0 | 1 | 2 > The parameter controls the pin supply style. Default: 0 (Do not output a `vddName`)

This variable 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.

0: No `vddName` will be output (Default)

1: Behavior depends on setting of `voltage_map`:

`voltage_map`=0: Use `related_power_pin`
`voltage_map`=1: Use `input/output_signal_level`

2: `vddName` will be output in the .libs following the format as specified by `voltage_map`.

This variable must be used before **char_library**.

power_add_input_pin

< 0 | 1 > Specify whether to include non-switching side input pin current. Default: 1 (include input pin current)

Set the variable to **0** to ignore the power contribution from non-switching side input pins regardless of the **pin_based_power** setting. When set to **1** (Default) Liberate will add non-switching side input pin power according to the **pin_based_power** settings of **1** or **2**.

This variable must be used before **char_library**.

power_adjust_for_pin_load

< 0 | 1 > Set this variable to remove the harness power from the internal_power. Default: 0

Set this variable to **1** to separate the power consumed by the pin_load in the harness from the cell internal_power. When set to **0** (default), the power will not be separated.

This command must be used before **char_library**.

power_binate_arc

<"separate" | "merge" > Splits the power into multiple groups based on a WHEN condition. Default: "merge".

Allows the power to be split into multiple groups based on a WHEN condition. Used when multiple timing tables exist based on the timing_sense.

Set to **separate**: Creates state-dependent power tables.
Set to **merge**: Merges power into a single table that is not state dependent. (Default)

This command must be used before **char_library**.

power_combinational_include_output

< 0 | 1 > Specify whether to include output pin in when. Default: 1 (Include output)

Controls whether the output pin is included in combinational cell leakage and hidden power `when` conditions.

0: Excludes the output pin from the `when` state.
1: Includes output pin in leakage `when` states. (Default)

This command must be used before **char_library**.

power_info

< 0 | 1 | 2> Print additional messages regarding power calculations. Default: 0 (no extra information)

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

Note that some of the power calculations are done prior to **write_ldb**, while other calculations are performed after **read_ldb** or **write_ldb** but prior to **write_library**. Therefore, it may 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_log_filename for additional information.

This feature is not supported for distributed jobs. It is only meant to be run locally for debug purposes.

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.

This command must be specified before **char_library**.

power_info_log_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 variable must be specified before **char_library** for all calculations.

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. (Default)
1: Disables modeling of `leakage_power` and `internal_power` for rails having 0v.

This command must be used before **write_library**.

power_multi_output_binning_mode

< 0 | 1 > Control binning methodology for cells with multiple output pins. Default: 0

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 (Default)
1: Bin for both timing and power effects (Recommended)

This variable must be used before **char_library**.

power_sequential_include_complementary_output

< 0 | 1 > Specify whether to include a complementary output pin in when. Default: 1 (Include output.)

Use this parameter to control whether a complementary output pin is included in sequential cell leakage and hidden power *when* conditions. Default: **1** (Include output pin in leakage *when* states). Set this variable to **0** to exclude the complementary output pin from the *when* state.

This command must be used before **write_library**.

power_sim_estimate_duration

< 0 | 1 | 2> Allows the user to specify a different method for the power tend. Default: 2

This variable follows the `tran_tend_estimation mode` for consistency.

0: not implemented.
1: estimate the power tend using the slew measurement threshold.
2: used in 12.1 ISR3 and earlier behavior. This setting is for backward compatibility.

power_subtract_leakage

< 0 | 1 | 2 | 3 | 4 > This parameter controls subtracting leakage from internal power. Default: 1

This determines what method is used to subtract leakage power from the energy numbers in the internal_power tables. (See Leakage Power for more information.) Supported modes are:

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 that an incomplete set of leakage_groups may 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.

This variable must be used before **char_library**.

power_subtract_leakage_msg_level

< 0 | 1 > Enable messages for state matching during leakage subtraction. Default: 0 (Off)

This variable 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.

0: No warnings reported, even when mismatches occur. (Default)

1: Warnings for mismatched or missing leakage states reported. (Recommended)

This variable must be used before **char_library**.

power_subtract_leakage_mode

< 0 | 1 | 2 > Specifies mode for subtracting leakage from power tables.
Default: 1

This parameter controls the methods Liberate uses to subtract leakage power from internal_power tables.

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. (Default)

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. (Recommended)

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}
```

This variable must be used before **char_library**.

power_subtract_output_load

<none | all | one | one_rise_full | one_both_half | all_rise_full | all_both_half>
Specify method used to subtract the output load.
Default: all (Subtract output load energy for all switching outputs from internal power.)

This parameter can be used to direct Liberate how to subtract the output load from the internal power. It supports the following values:

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 command must be used before **char_library.**

power_subtract_output_load_mode

< 0 | 1 | 2 >	Specify to use final simulation output voltage or the supply rail value. Default: 0 (Use the simulation final output voltage) Recommended: 2

When computing the switching `internal_power`, Liberate will subtract the power used to drive the output load. Liberate will subtract CVV or 0.5CVV depending on the setting of the **power_subtract_output_load** variable. The C in CVV is the output load capacitance.

Use this variable to determine the values represented by the Vs in the CVV.

0	Alspice: `0.5*C*Vsupply*Vfinal(alspice)` extsim: `0.5*C*Vfinal*Vfinal`
1	`0.5*Vsupply*Vsupply`
2	`0.5*Vsupply*Vfinal` (All simulators)

This variable must be set before **char_library**.

power_tend_match_tran

< 0 | 1 > Force power_tend to match tran_tend in SPICE simulations when using external simulators (see char_library -extsim) without the SKI interface. Default: 0 (Do not force a match).

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 that trapezoidal ringing in the current waveforms lead to both inconsistent and incorrect power results. Setting this variable 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).

0: Do not match power_tend to tran_tend (Default).

1: Match power_tend to tran_tend (Recommended).

Note: 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 variable 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 variable 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 variable must be set before **char_library**.

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.)

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.

0: Disabled (Default)
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.

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 variable is disabled when the **predriver_waveform_ratio** variable is set to 0.
2. For library validation (LV) if a normalized wavewform exists in the library, it will override the predriver_waveform setting.

This variable must be used before **char_library**. Example:

```
# Use a PWL pre-driver derived from an RC network
set_var predriver_waveform 2
```

predriver_waveform_mode

< 0 | 1 >	When set to 1, force the delay and transition measurement thresholds to be included in the predriver waveform. Default: 0 (Use best curve-fit.))

This parameter instructs Liberate to include the exact delay (see measure_delay*) and 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 delay and slew measure threshold values.

0: Use best curve fit result (Default)
1: Include the exact delay and slew measure threshold values in the input waveform. Use curve-fitting to determine the remaining points. This is the recommended setting because it removes the interpolation errors of the input waveform from the delay measurements.

This variable must be used before **char_library**.

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 variable 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 variable 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 variable must be used before **char_library**. 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: 1 (preserve the function)

Liberate has the ability to generate a logic function from basic building blocks: AND, OR, INV. This is used during Verilog and Vital modeling.

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. (Default and recommended.)

prevector_period

<value> Set the period used by the prevector. Default:1e-8 (10ns)

This variable sets the period used by the **define_arc -prevector** option. Each vector listed in the prevector will be simulated for the time period specified by this variable.

This variable must be used before **char_library**.

prevector_slew

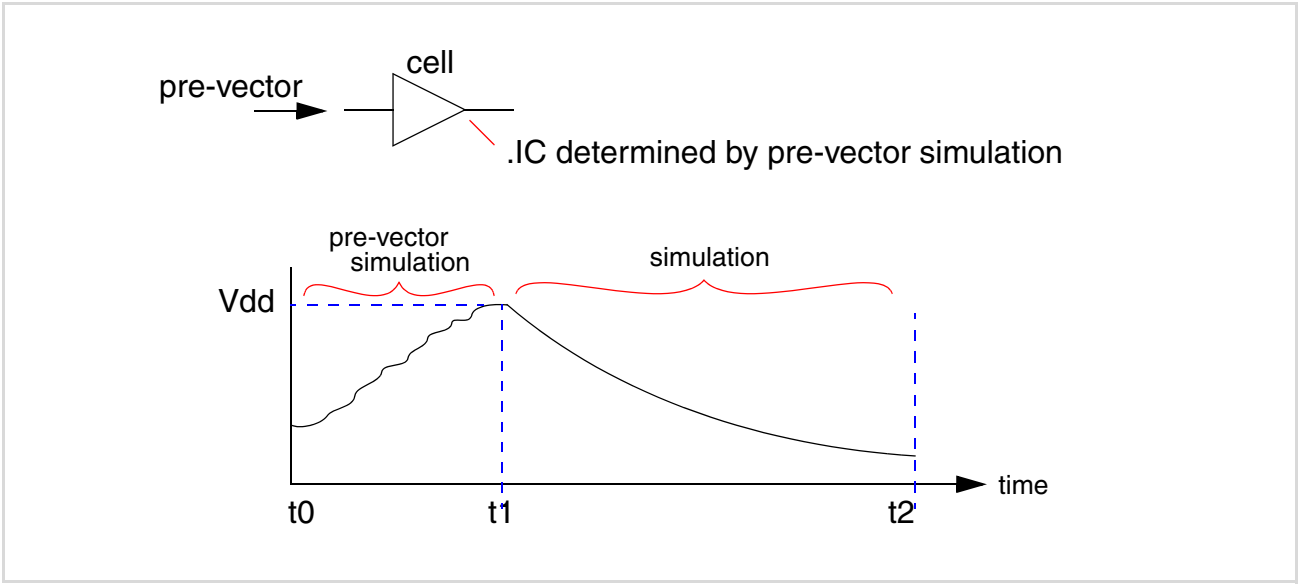
<value> Set the transition used by the prevector. Default: 1e-10 (100ps)

This variable sets the transition (slew) used when pins in the **define_arc -prevector** transition from one state to another.

This variable must be used before **char_library**.

prevector_voltage_waveform_mode

< 0 | 1 | 2 > Selects which part of the pre-vector waveform will be stored in the Library. Default: 1



0: Store waveform from **t0** to **t1** ("Incorrect" behavior, prior to release 3.2)
1: Store waveform from **t1** to **t2** (Default and Recommended)
2: Store waveform from **t0** to **t2** (Might be needed for some analog circuits.)

This variable must be set before **char_library**.

process_match_pins_to_ports

< 0 | 1 > Set this to skip cells with define_cell/subckt port mismatch. Default: 0

If this parameter is set to a 1, Liberate will output an error and the cell will be skipped when there is a mismatch between the define_cell pin definitions and the subckt ports. The default is 0 (give warning on mismatch and continue).

This variable must be used before **char_library**.

ramp_vsrc

< 0 | 1 > Enables supply ramping. Default: 0

Set this variable to enable supply ramping. This can help with DC convergence related issues when using an external simulator. When set to 1, this variable will tell Liberate to always ramp

the supply. This ramping will be enabled for all data acquisition and will override the **leakage_ramp_vsrc** variable. The power supply will ramp from 0s to ½ sim_init_duration.

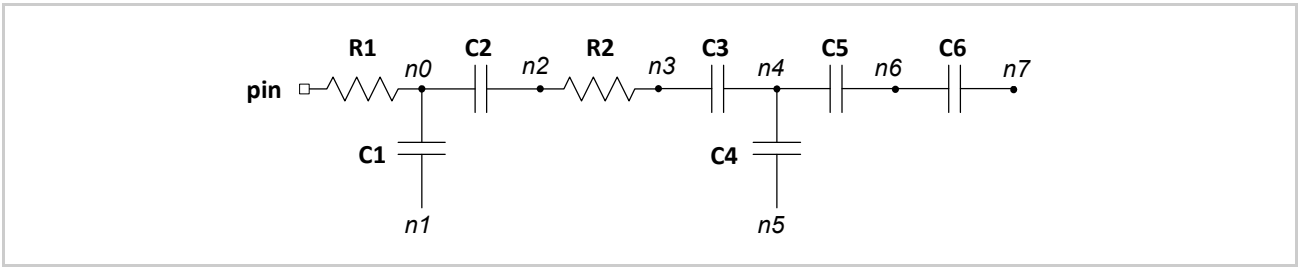
This variable must be used before **char_library**.

rc_floating_cap_mode

< 0 | 1 | 2 | 4 > Controls the handling of floating-cap nodes. Default: 1

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.)



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. (Default)

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.

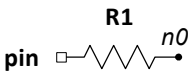
For Spectre, the file contents will look like this:

```
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. Using the example above, the result will be this:



This variable must be used before **char_library**.

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 variable must be used before **char_library**.

rdb_checkpoint_dir

<directory_name>	Full path to the directory where the rdb checkpoint file will be 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 can be useful when characterizing extremely large IO cells that have very long runtimes. 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 variable.

This variable must be set before **char_library**.

Example:

```
set_var rdb_checkpoint_dir /NFS/work/rundir/RDB
```

rdb_exit_if_source_differ

< 0 | 1 >	Exit if mismatched PVT corners are found.

The RDB flow is enhanced to ensure that exactly the same script/settings are used before restoring characterized values from RDB. This ensures data consistency is maintained from run to run even if the user errorneously sets the rdb_checkpoint_dir variable to the same location.

The default and recommended value is 1.

This variable must be used before **char_library**. 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 variable 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 variable must be used before **char_library**. 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 variable is not set, then removal constraint characterization will follow the constraint_glitch_peak value.

This variable must be used before **char_library**. Example:

```
# Set removal glitch peak to 5%
set_var removal_glitch_peak 0.05
```

res_merge

< 0 | 1 > Merge resistors to improve simulation time. Default: 1

Allows Liberate to merge resistors to improve simulation run time. Default is 1 (on.) Set to 0 to disable.

This variable must be used before **char_library**.

res_open_tol

<value> Controls filtering out resistors larger than this variable's setting.

The **res_open_tol** variable lets you filter out (and leave the connection open) any resistor that is larger than what this variable is set to, only if **res_open_tol** is set to a value larger than 0.

This variable must be used before **char_library**.

res_tol

<value> Threshold value to short resistors. Default: 1.0

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 be 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 (e.g. resmin=0.001 in HSPICE).

This variable must be used before **char_library**.

reset_leakage_current_mode

< 0 | 1 > Turn any negative leakage value to zero. Default: 0 (Zero negative leakage for Library Compiler compliance)

Controls when leakage will be reset to 0 (zero) amps.

0: Reset leakage current greater than 0 amps to 0 for `primary_ground` or `backup_ground` pg pins. Reset leakage current less than 0 amps to 0 for all other pg pins. (Default and recommended)

1: Do not reset leakage current when the `pg_type` is `primary_power`, `primary_ground`, `backup_power`, or `backup_ground`.

This variable can be used after **char_library**.

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 variable will not be applied to mpw constraints.

This variable can be used after **char_library**.

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 may be used after **char_library**. Examples:

```
# 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 > Turn any negative power sum to zero (Default: *0*)

Set this variable to **1** to avoid using negative leakage currents in power calculation when the leakage has been zeroed out for LC compliance. The recommended setting is **1**.

When the variable `reset_negative_power` is set to `1`, the variable `reset_negative_leakage_power` will have no effect and all negative power, including leakage will be reset.

This variable must be set before **char_library**.

reset_negative_power

< 0 | 1 | 2 > Turn any negative power sum to zero. Default: 1

By default, Liberate will reset 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, ie: rise vs. fall.) The sum of corresponding entries between rise/fall pairs is checked, and if it is less that zero (e.g., 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.

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, they both will be set to zero. *See algorithm above*. (Default)

2: Any power that is negative will be set to zero.

This variable must be used before **write_library**.

resolve_collision

< 0 | 1 | 2 >	Set to have Liberate always resolve collision vectors with the requested spice engine. Default: 1

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 will result in a digital behavior. Set this variable to **2** to use Alspice to resolve collisions even when **extsim_model_include** is enabled. When set to **1** (default), collisions will be resolved using the external simulation engine when **extsim_model_include** is used. A setting of **0** will disable collision resolution using simulation.

This variable must be set before **char_library**.

retry_count

<value>	Specifies the number of retries for a particular arc. Default: 1

This variable lets you specify the number of retries for a particular arc, pausing 5 seconds between retries. A "1" instructs Liberate to retry failing simulations one time. (Default)

This variable must be set before **char_library**.

retry_count_file_operation

<value>	Controls the number of retries when renaming, moving, or copying a file. Default: 10

This variable lets you specify how many times to retry when an attempt to move, copy, or rename a file fails due to network or LSF issues. The default value is 10 and a valid value is an integer between 0 and 60, both inclusive.

This variable must be set before **char_library**.

rsh_cmd

<ssh | rsh>	The shell command to use for accessing a remote client when using distributed parallel processing. Default: ssh

The above parameters are used to control the interface to remote clients when using distributed parallel processing. Before using parallel processing, make sure that the server machine (the machine Liberate is to be invoked from) can perform either an **rsh** or an **ssh** and an **rcp** or a **scp** to each client machine without requiring a password or passphrase. The **rsh_cmd** command string can reference the current client and the command to invoke Liberate with by using **%M** (machine) and **%C** (command). (See Distributed Processing for list of all "percent" (%) variables.) In addition, command line options that appear after the Liberate Tcl file name can be passed into the **rsh_cmd** string by using **%O** (options). These substitutions can be useful if your system is using a job queuing system.

This variable must be used before **char_library**. Examples:

```
# Set the shell and copy variables for distributed runs
# set remote file copy to rcp instead of scp
set_var rcp_cmd rcp
# set remote shell to rsh instead of ssh
set_var rsh_cmd rsh
```

scale_load_by_template

< 0 | 1 >	Set this parameter to scale the load indices using the values defined in the template when generating indices using the **auto_index** argument of **char_library**. Default: 0 (off)

This parameter turns on scaling of the load indices created by the **auto_index** option of **char_library**. Normally, when the **auto_index** option of **char_library** is enabled, the intermediate indices are determined using a geometric sequence. When this control variable is set, the indices 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 variable must be used before **char_library**.

scale_tran_by_template

< 0 | 1 > Set this parameter to scale the transition indices using the values defined in the template when generating indices using the **auto_index** argument of **char_library**. Default: 0 (off)

This parameter turns on scaling of transition indices created by the **auto_index** option of **char_library**. Normally, when the **auto_index** option of **char_library** is enabled, the intermediate indices are determined using a geometric sequence. When this control variable is set, the indices 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 variable must be used before **char_library**. 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. Default: **0** (do not output leakage power in the scan dummy)

This variable must be used before **char_library**.

sdf_cond_equals

<string> *sdf_cond* attribute style (Default: "")

Use this variable to specify the format for the *sdf_cond* construct to be used in output models such as Liberty, Verilog, and Vital models. This variable supports the following values: "**==**", "**== logical**", "**== binary**", "**===**", "**=== logical**" and "**=== binary**". This variable is identical to and will replace the **sdf_cond_equals** argument 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 variable such that any subsequent **write_verilog** or **write_vital** commands will also have this modification. This variable is sequence dependent. It only impacts the models written after setting this variable.

This variable can be used after **char_library**. 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_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 (i.e. it has two or more operands) it will be replaced by a name of the form *sdf_cond_prefix#* where the **#** is unique for each **sdf_cond** within the cell. The **sdf_cond** attributes are used for writing conditional timing arcs in Liberty, Verilog and Vital formats (see **write_library**, **write_verilog and write_vital**).

This variable must be used before **char_library**. Example:

```
# Set the sdf_cond prefix
```

```
set_var sdf_cond_prefix "int_cond"
char_library
write_library my.lib
```

sdf_cond_style

< 0 | 1 > Specify the **sdf_cond** style. Default: 0

The sdf_cond attribute for complex conditional constraint arcs must be represented as a "variable" that is the functional equivalent to the *when* condition. When set to **0** (Default), Liberate will generate a unique variable for each different condition as follows: "${sdf_cond_prefix}#" where the **sdf_cond_prefix** is set using the "sdf_cond_prefix" variable (default: **adacond**). If sdf_cond_style is set to **1** a variable name will be created from the "when" condition such that the logic *and* operators will be replaced with **AND**, logic *or* with **OR**, logical *not* with **NOT**, ***(*** with **OP**, and ***)*** with **CP**. These operands will be separated from the operators by the character: _. This "constructed" variable name will be prefixed with the value of the variable **sdf_cond_prefix**. Example:

```
"A * (B | !C)" becomes "adacond_A_AND_OP_B_OR_NOT_C_CP"
```

Note that **sdf_cond_style 1** applies to all conditional constraints while sdf_cond_style **0** only applies to complex *when* conditions that have multiple operands.

This variable can be set after **char_library**.

sdf_logic_and

"string" The characters to use for denoting logic AND in **sdf_cond** attributes. Default: " & " *(Notice the space on both sides of &)*

This parameter sets the logic AND string for **sdf_cond** attributes.

This variable must be set before **char_library**.

sdf_logic_not

"string" The characters to use for denoting logic NOT in **sdf_cond** attributes. Default: "~"

This parameter sets the logic NOT string for **sdf_cond** attributes.

This variable must be set before **char_library**. 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"	The characters to use for denoting logic OR in **sdf_cond** attributes. Default: " | " *(Notice the space on both sides of |)*

This parameter sets the logic OR string for **sdf_cond** attributes.

This variable must be set before **char_library**.

server_timeout

<value>	Specify unresponsive client message interval. Default: 10800 (seconds)

When the server is waiting for a response from a client, it will print out a message when it has waited for the timeout specified by this variable. The message will repeat every timeout duration.

This variable must be used before **char_library**.

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. Please check the simulation status, for example, check sim.lis file if using external simulator.

sim_duration

<value>	The maximum simulation duration. Default: 1e-7

Set this variable to the maximum allowed simulation time to be used during spice simulation. This variable puts a limit on the total simulation duration. It acts as a global override. Default: 1e-7

This variable must be used before **char_library**.

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 variable to disable the algorithm in Liberate that will estimate the expected simulation duration. Set this variable to **0** to disable the estimation. Once disabled, the simulation will run for exactly the time specified by the control variable **sim_duration**.

This variable must be used before **char_library**.

Example:

```
# Set the parameter 'sim_duration' to 1e-5 seconds
set_var sim_estimate_duration 0
```

sim_init_condition

<hybrid | ic | ic_except_dc> Specify initial condition method. Default: hybrid

Set this variable to tell Liberate how to set initial conditions when presenting spice decks to the simulator. 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 **hybrid** option is the default. When set to **hybrid**, Liberate will use *.ic* for any floating nodes and .nodeset everywhere else.

This variable must be used before **char_library**. Example:

```
set_var sim_init_condition ic_except_dc
```

sim_init_condition_estimation_mode

< 0 | 1 > Specify initial condition method. Default: 1

Use this variable 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 thru pmos and nmos channels. In the second pass, Liberate will propagate vdd/gnd thru all channels for those nets that didn't get set in the first pass. (Default: **1**). To revert to release 2.3p2 and prior release behavior, set this variable to **0**.

This variable must be used before **char_library**.

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. This parameter will only have an effect when used in combination with sim_use_init_duration.

When the ramp_vsrc variable is set, the sim_init_duration variable will determine the amount of time Liberate will simulate to determine the internal node voltages. If ramp_vsrc is set, then this variable must also be set. The leakage_sim_duration variable will override this variable for all leakage simulations.

This variable must be used before **char_library**. 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 variable 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 variable must be used before **char_library**. 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 variable 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 variable must be used before **char_library**.

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 variable 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 variable `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 variable must be used before **char_library**. For example:

```
# Enable simultaneous switching analysis.
set_var simultaneous_switch 1
```

simultaneous_switch_from_cell_when

< 0 | 1 > Enables automatic dual input switching. Default: 0

When a cell has a set of inputs with logic relationships between them, such as one_hot or dual inputs, Liberate can automatically find one or more arcs with simultaneous switching inputs (up to 2 inputs switching) that satisfy the `-when` argument of the **define_cell command** for each arc. You must specify the `-when` argument of the **define_cell** command to define the logical relationships of the cell pins. This feature is currently limited to one or two switching inputs for each of the related_pin and constrained_pin.

0: Do not look for simultaneous switching side pins.
1: Look for simultaneous switching side inputs such that the **define_cell** and **define_arc** -when states will be met both before and after all of the pins switch.

Two commands are available to assist with adding the when states to the **define_cell** command. They are: **one_hot** and **one_cold**.

This variable must be used before char_library. For example:

```
define_cell -when "ck^ckb" ...
# Enable simultaneous switching analysis.
set_var simultaneous_switch_from_cell_when 1
```

simultaneous_switch_offset

<value>	Offset to be used between the related_pin and the simultaneously switching side inputs. Default: 0 (seconds)

Set this variable 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 variable must be used before **char_library**. For 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*)

This variable will define how the `-when` argument of the **define_arc** command is applied:

- if `simultaneous_switch_from_cell_when` is enabled (1 or 2).
- if `simultaneous_switch_from_cell_when` is 0, this variable has no effect.
- if `simultaneous_switch_from_cell_when` is enabled, then this variable 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 variable is 0.

This variable must be used before **char_library**. 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 (use mid-point of slew/load indices for binning).

Set this variable to 4 to revert to release version 2.3 (and earlier) behavior where 4 slew/load simulations were used. These 4 slew/load simulations are min_slew/min_load, min_slew/max_load, max_slew/min_load and max_slew/max_load.

This variable must be used before **char_library**. Example:

```
# Use 4 slew/load simulations for binning worst simultaneous
# switching vectors
set_var simultaneous_switch_worst_vector 4
```

ski_clean_mode

< 0| 1 | 2> Enables clean-up of the shared memory environment used by SKI. Default: `0` (not enabled).

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 use the standard Spectre interface. While this yields accurate results, there is a performance penalty.

0: Perform no clean-up. (Default)
1: Perform standard cleanup via adding "-a" to the clean-up command. (Recommended)
2: Same as 1, but log additional debugging information via adding "-d" to the clean-up command. This is useful to debug situations where SKI is consistently being disabled on certain farm machines.

Because these resources follow the standard rules of Unix permissions, a user cannot clean up resources belonging to another user. The resources must either be cleaned up by the owner or by root.

This variable must be used before **char_library**.

ski_compatibility_mode

< 0| 1 > Enables consistency between standalone Spectre and Spectre-SKI. Default: 0.

0: Each individual SKI control variable is set manually.
1: Set the following variables to 1

- ski_power_subtract_output_load_match_extsim
- alspice_power_subtract_output_load_match_extsim
- ski_alter_mode
- ski_mdlthreshold_exact
- capacitance_save_mode

The dependent variables listed above are under Liberate control and cannot be overridden while `ski_compatibility_mode` is set to 1.

This variable must be used before **char_library**.

ski_enable

< 0 | 1 > Enables SKI-based interface in Liberate. Default: 0 (off).

This variable 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.

0: Disables the Spectre high-performance (SKI-based) interface. (Default)
1: Enables the Spectre high-performance (SKI-based) interface.

Important:

- Requires Spectre version **MMSIM 10.1 ISR20**, or **MMSIM 11.1 ISR8**, or equivalent newer versions.
- To use Spectre as the circuit simulator, the "**char_library -extsim spectre**" option must be set.
- SKI is only supported in the parallel packet flow.

This variable must be used before **char_library**.

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 will use a setting similar to Spectre's `mdlthresholds=interpolated` behavior. (Default)

1: SKI will use a setting similar to Spectre's `mdlthresholds=exact` behavior. (Recommended for consistency with standalone Spectre default behavior). For more details and additional settings, see the Correlation between stand-alone Spectre & SKI section.

This variable must be used before **char_library**.

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)

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.

0: SKI will use charge-based power calculation. (Default)
1: SKI will use voltage-based power calculation. (Recommended)

This variable must be set prior to **char_library**.

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 variable will force a memory reset after the specified simulation iteration count. This variable 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 variable only has an effect if **ski_enable=1** (Spectre-SKI is enabled). See External Simulator Options and Settings.

This variable must be used before **char_library**.

slew_lower_fall

<value> The % point on the cell output waveform to output falling output transition times to. Default: 0.3 (30%)

slew_lower_rise

<value> The % point on the cell output waveform to output rising output transition times from. Default: 0.3 (30%)

These variables must be used before **char_library**.

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 a **0**.

This variable must be used before **char_library**.

slew_upper_fall

<value> The % point on the cell output waveform to output falling output transition times from. Default: 0.7 (70%)

slew_upper_rise

<value> The % point on the cell output waveform to output rising output transition times to. Default: 0.7 (70%)

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_*** variables (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_*** variables (x 100).

These variables must be used before **char_library**. Examples:

```
# 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 1 to enable sorting pin timing groups by "when" alphabetically. Default: 0 (Do not 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 will not be sorted alphabetically but instead follow an arbitrary order. This command is useful when a deterministic order is desired for the timing groups under a pin.

This variable 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: 0 (don't sort.)

Pins for each cell may be sorted in forward or reverse alphabetical order.

0: Don't sort pins. (Default)
1: Sort pins in alphabetical order.
2: Sort pins in reverse alphabetical order.

This variable can be used together with pin_type_order, to specify that pins be sorted within groups, ie: keep all input pins together and sort them; keep all output pins together and sort them; etc.

This variable must be used before **write_library**.

sort_pins_under_when

< 0 | 1 > Enable sorting of pins in a WHEN. Default: 0 (do not sort alphabetically)

Liberate can sort the pins in a *when* condition in the output library. Default: **0** (Do not sort pins). When set to **1**, Liberate will revert to 2.0p3 behavior of sorting pin names within the *when* string as follows:

```
input pins - alphabetical, followed by
bidi pins - alphabetical, followed by
output pins - alphabetical
```

This variable must be used before **char_library**. Example:

```
# Sort the pins
set_var sort_pins_under_when 1
```

spectre_dash_log

< 0 | 1 > Enables the Spectre `-log` command line option when set to 1. Default: 1.

Set this variable 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 variable must be set before **char_library**.

spectre_use_char_opt_license

< 0 | 1 > Controls use of `Spectre_char_opt` license feature. Default: 1 (use the feature)

This variable enables a lower-cost option to use Spectre for characterization only. This option *must* be purchased and your license file *must* contain the `Spectre_char_opt` feature.

Important

If your license file does not contain the `Spectre_char_opt` feature, you should set `spectre_use_char_opt_license` to 0. This will prevent a slow down in performance due to Spectre checking for a nonexistent license feature.

0: Do not use the `Spectre_char_opt` feature.
1: Use the `Spectre_char_opt` feature. (Default)

This variable must be set before **char_library**.

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 variable 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 (e.g. &, |) with Spectre.

This variable must be used before **read_spice**.

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 variable 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 variable is treated as a hierarchical delimiter. In the SPICE decks that are written out, the first character in this string will be used as the hierarchical delimiter.

This variable must be used before **char_library**. 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 variable to specify the character that will be used to replace those delimiter characters. Default "_".

This variable must be used before **char_library**. 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 variable off to allow netlists with instance names not prefixed by "X".

This variable must be used before **read_spice**. 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 netname to the shortest node name, Liberate will set the logical wire name using the name of the attached instance.

In cases where a VDB flow is used with switch cells and multiple netlist extractions, setting this variable will resolve a certain class of errors. The recommended value is 1.

This variable must be used before **char_library**.

subtract_hidden_power

< 0 | 1 | 2 >	Control the subtraction of hidden power from internal power. Default: 0 (Do not subtract hidden power.)

Use this control variable to cause hidden power to be subtracted from internal_power.

0: Do not subtract hidden power. (Default)
1: Subtract hidden power in sequential cells.
2: Subtract hidden power for all cells.

This variable must be used before **char_library**.

subtract_hidden_power_use_default

< 0 | 1 | 2>	Enables subtraction of the default hidden power, if available. Default: 0 (Disable hidden power subtraction)

Use this variable to control what value of hidden power should be used when subtracting hidden power from switching internal_power.

0: (Default) 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** variable 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.

This variable will have no effect if **subtract_hidden_power** is set to `0`.

This variable must be set before **write_library**.

supply_define_mode

< 0 | 1 > Select algorithm for determining supplies. Default: 0

When set to **1**, Liberate will determine the supplies from module wires rather than tracing transistor connectivity. When set to **0** (default), Liberate will determine the supplies by tracing the transistor connectivity. A setting of **1** can be useful in matching legacy libraries.

This variable must be set before the **set_operating_condition** command.

This variable must be used before **char_library**.

switch_cell_bounded_dc_current

< 0 | 1 > Enable bounding the dc_durrent sweep between supplies. Default: 0 (Do not bound.)

Set this variable to limit the dc current sweep for power switch cells. When this variable is set to **0**, the dc current sweep reported for the power switch cells will use the same voltage range as the CCSN dc_current. When this variable is set to **1**, the dc_current sweep will be limited to the gnd and vdd rail voltages.

This variable must be used before **char_library**.

switch_cell_dc_current

< 0 | 1 > Include/omit dc_current from switch cells. Default: 1 (Include dc_current.)

0: Omit the **dc_current** from switch cells. This can be useful when using a separate run to characterize leakage to filter the **dc_current** from the library for switch cells.
1: **dc_current** will be included in the switch cells when **define_cell -internal_supply** is provided. (Default)

This variable must be used before **char_library**.

switch_cell_dc_current_output_offset

<value> Specify voltage offset value used in switch cell dc_current. Default: 0

Use this variable 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 variable must be used before **char_library**.

switch_cell_powerdown_function

< 0 | 1 > Specifies whether to generate the powerdown_function on switch cells. Default: 0

Setting **switch_cell_powerdown_function** to 1 will generate the **powerdown_function** on switch cells.

This variable must be set before **char_library**.

template_unique_power_mode

< 0 | 1 > Default: 0

Note: This variable should never be set manually. It is automatically set 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 may 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 variable is set to 1, the above issues are addressed, resulting in correct power values and runtimes as fast as without `-unique_power` argument.

This variable must be set before **char_library**.

test_cell_at_end

< 0 | 1 > Control the position of the test_cell.
Default: 0 (do not move the test_cell)

Set this variable to **1** to 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 variable must be set before **write_library**.

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 variable applies to timing groups that are not default timing groups. For control of merging for default groups, see the define_arc_ignore_mode variable.

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`.

Note: Setting this variable to `separate` can *adversely impact* model generation for all define_arc/verbose flows.

This variable must be set before **char_library**.

tmpdir

<string> Specify a temporary working directory. Default: (none)

This variable specifies a temporary working directory for Liberate to store extsim run data.

Either the Tcl variable `tmpdir`, or the environment variable `TMPDIR` can specify a temporary working directory. If both variables are set, the Tcl variable `tmpdir` takes precedence. Liberate follows this rule 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 variable 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 variable is set to 1, the *when* specified in the leakage groups will not include the output pin state.

This variable 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 > Switch between different algorithms for estimating the transient simulation end time.

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 tran_tend using the slew measurement threshold.

tristate_disable_transition

< 0 | 1 > Output disable transition values in the library.
Default: 0 (output scalar value)

Use this parameter to request Liberate to output full transition tables for the tristate disable arcs. By default, Liberate will output a scalar value for the *three_state_disable* arcs.

This variable can be used after **char_library**.

tristate_pin_cap_always_on_res_mode

< 0 | 1 > Corrects the affect of adjust_tristate_load when pin capacitance is very large. Default: 1 (on)

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.

0: Behavior of release 12.1.1 and earlier.
1: Corrects the effect of adjust_tristate_load when pin capacitance is very large. (Default)

This variable must be set before **char_library**.

use_pid_tmpdir

< 0 | 1 > Add the process ID to the tmp directory name.
Default: 1 (Add pid)

Add the process ID to the tmp directory name. (Default.). Examples:

```
/tmp/<user>/altos.<pid>.0
/tmp/<user>/altos.<pid>.1
```

This variable must be used before **char_library**.

user_data_attr_order

< 0 | 1 > Request user_data attribute order. Default: 0

Set this parameter to control 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: 0 (Follow default Liberate order).

This variable can be used after **char_library**.

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 variable 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: max_capacitance (a simple attribute under the pin() group), cell_leakage_power (a simple attribute under the cell() group), voltage_map and pg_pin.

The built-in default list of attribute overrides currently includes:

```
{area function three_state state_function}
```

This variable can be used after **char_library**. 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 variable must be used before **write_library**.

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 variable to 0 for 2.5p2 and prior behavior.

This variable should be used before **char_library**.

vector_estimate_dump

< 0 | 1 > Prints the result of the estimation decks for use in vector debugging. Default 0 (Off).

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, then the results of the full simulation should be reviewed.

0: Specifies not to print vector estimate information. (Default and recommended)

1: Prints vector estimate information intended for template debugging purposes.

This variable must be set before **char_library**.

vector_side_input

< 0 | 1 > Set unknown side inputs to 0 or 1.
Default: -1 (expand all X)

This parameter can be used to force side inputs whose values are not specified in a "when" or vector to all zeros or all ones. By default, Liberate will automatically decide what to set the side inputs to. This includes leaving them floating. It is recommended to always set all side inputs. Set to 0 for 0:X and set to 1 for 1:X.

This variable must be used before **char_library**.

verilog_cg_filter_edge

< 0 | 1 > Enable filtering of extra timing constraints in Liberty and Verilog.
Default: 1

If this variable 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**

The default is 1. Set this variable to 0 to disable this filter.
This variable must be set before **write_library** or **write_verilog**.

voltage_map

< 0 | 1 | 2 > Generate a voltage map in the output library
Default: 0 (off).

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.

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*).

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 variable must be used before **char_library**.

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 variable must be used before **char_library**.

waveform_report

< 0 | 2 > Enable saving of driver waveform. Default: 2

Set this control variable to 0 to disable the saving of the driver input waveform into the .vdb file (see **write_vdb**). Do not use this variable if the **write_library driver_waveform** option is to be used. Default: 2 (The driver input waveform will be saved into the .vdb and the .ldb).

This variable must be set before **write_vdb** and/or **char_library**.

wnflag

< 0 | 1 > Instructs Liberate to use W/nf for model selection. Default: 1

If the user has **.param wnflag=0**, they must set the variable **wnflag** to **0** in their Tcl run script to tell Liberate to use W *instead* of W/nf to select model binning.

0: Use W for model selection.
1: Use W/nf ratio for model selection. (Default)

Example:

```
set_var wnflag 0
```

This variable must be set before **read_spice**.

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. (Default)
1: Forces Liberate to read the ldb on disk prior to writing the library. Used to address possible precision issues between the `char_library` and `read_ldb`/`write_library` flow. (See also write_library **`-sync_ldb`**.)

This variable must be set before **write_library**.

write_logic_function

< 0 | 1 > Enable/Disable logic function generation. Default: 1 (Enable)

Enables/disables writing of the **logic function** when using Inside View. 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 runtime.

0: Turn off logic function determination when using Inside View.
1: Automatically write function and three_state attributes. (Default)

This parameter must be used before **char_library**.

write_logic_function_mode

< 0 | 1 > Alternate algorithm for generating logic function. Default: 1 (on)

Enables an alternate algorithm for determining logic function, designed to minimize X-propagation. **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. (Default)

This variable must be set before **write_library**.

write_min_transition_attr

< 0 | 1 > Write `min_transition` pin attribute to library. Default: 0

0: Do not output `min_transition`. (Default)
1: Output `min_transition`.

This parameter must be used before **write_library**.

write_template_force_power

< 0 | 1 > Output power templates whenever there is characterized power data. Default: 1

The **write_template** command will output a reference to a power template for every cell that has any power data. There are cells (ex.:DECAP and Filler cells) that do not have any timing or internal power arcs, but that only have leakage. Liberate must have a define_cell -power option referencing a power template (see define_template) for these types of cells to be characterized. Set this variable 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 variable must be set before **write_template**.

6

Library Comparisons

This chapter describes the Liberate utilities for comparing libraries.

Liberate provides ways to easily 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> File generated by **compare_library –gui** option

The **lcplot** graphical comparison plots make it very 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.

Pull-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.

Graphical Library Comparisons

There are 3 styles of plots available: X/Y, Accuracy, and Errorbound.

Style: X/Y

The following graphical 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 will fall exactly on the 45-degree axis. Note: By default, this plot type will display all four data types: Timing, Power, Leakage and Capacitance.

An example is shown below:

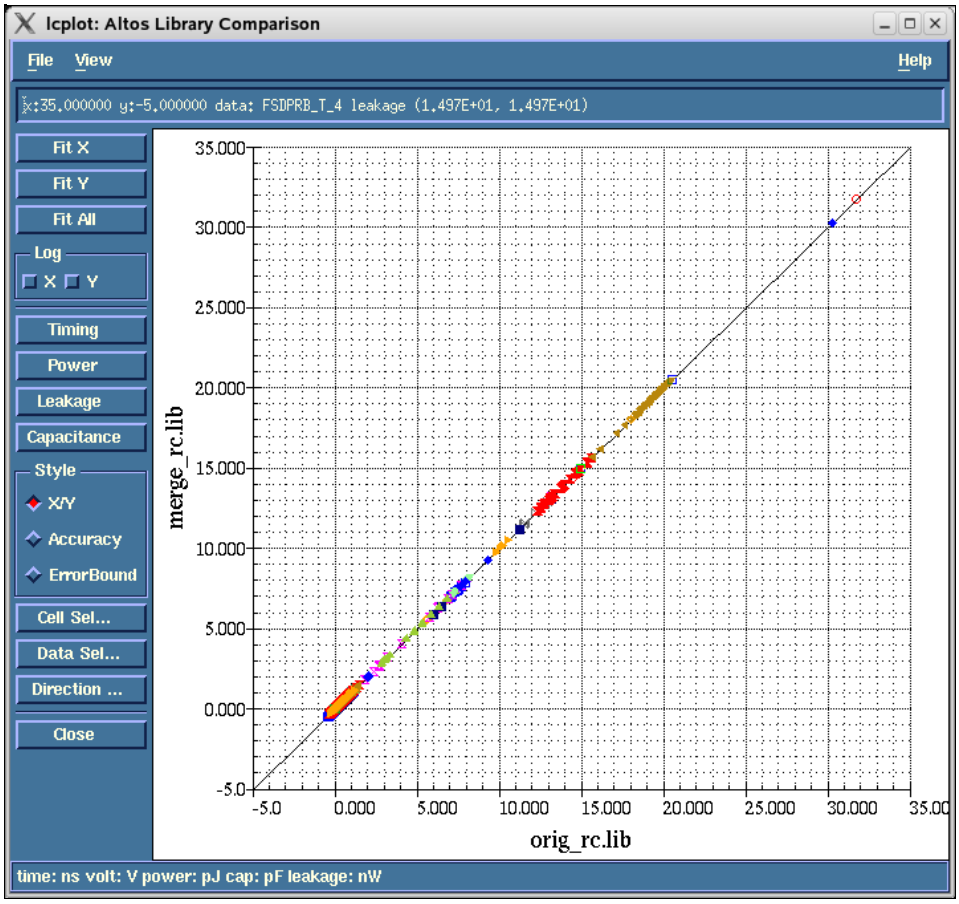


Figure 5: LCPlot GUI: X/Y Library Comparison

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 is shown below:

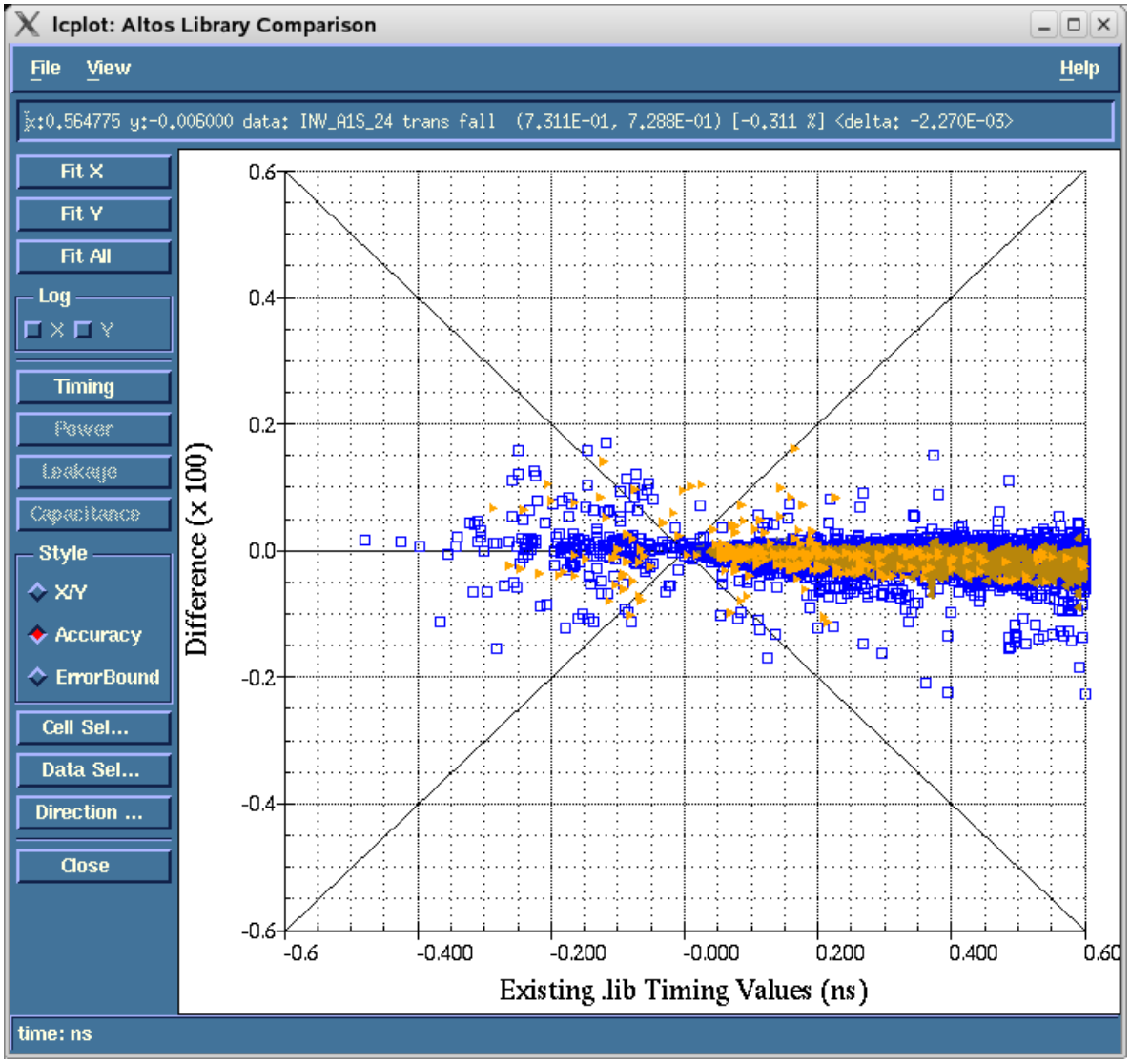


Figure 6: LCPlot GUI: Accuracy Library Comparison

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), then 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 is shown below:

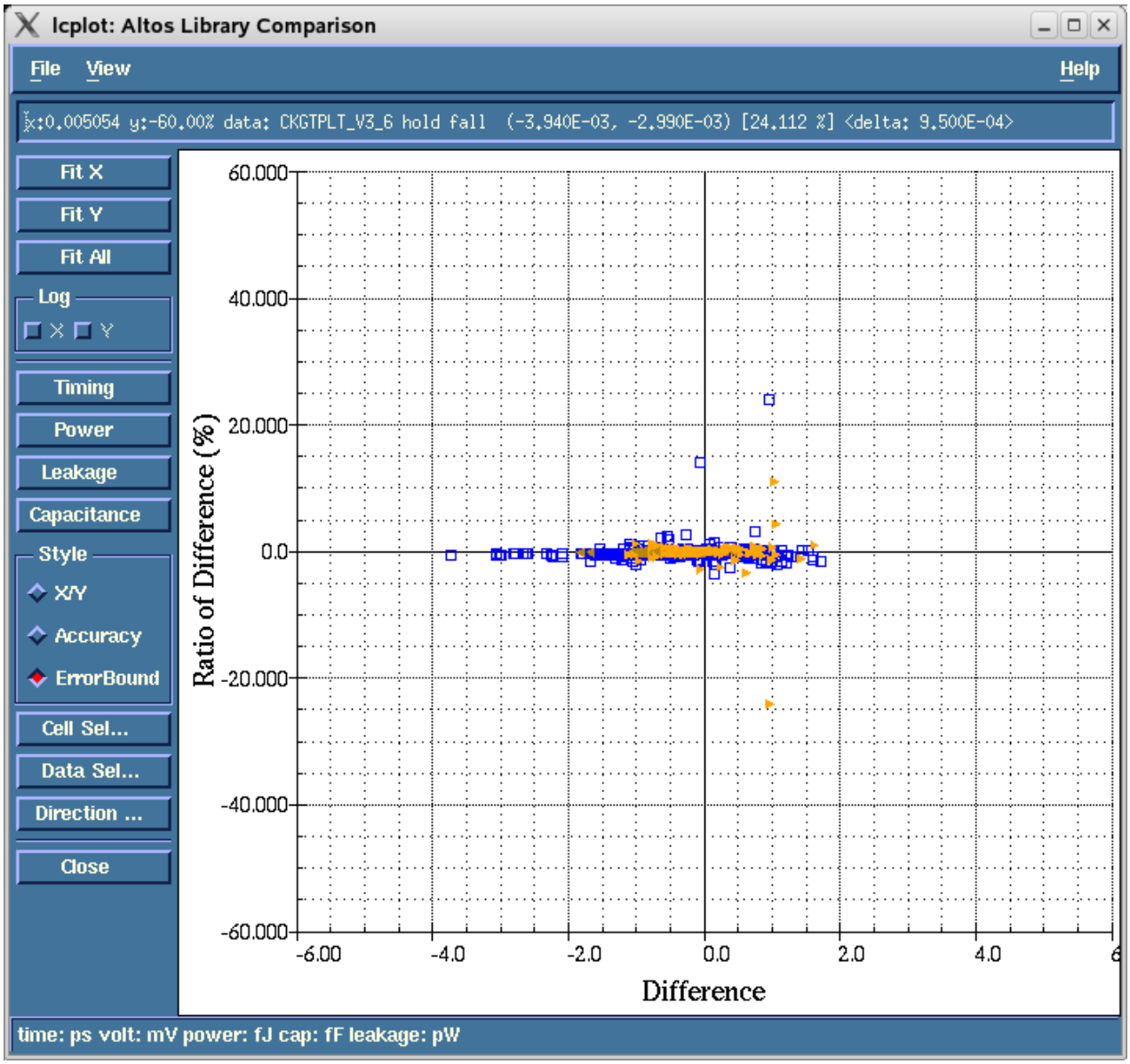


Figure 7: LCPlot GUI: ErrorBound Library Comparison

Data Selection

There are 2 ways to select data: Cell Type, and Data Type.

Cell Type Selection

To select the cells to compare, click on the **"Cell Select"** button. This pops up a selection menu that lists all of the cells in the library. An example is shown below:

Figure 8: Cell Selection Menu

To select a cell from the list click on the cell name and then click the **"Apply"** button. To select multiple cells hold down the **"Shift"** key, select the cell name followed by **"Apply"**. To add cells to the selected set hold down the **"Ctrl"** key, select the cell names and click on "**Apply**". To select all cells click the **"All"** button followed by **"Apply"**. To unselect all cells click on the **"None"** button followed by **"Apply"**. To quit the cell selection menu use the **"Close"** button.

Data Type Selection

To select the type of library data to compare, click on the **"Data Select"** button. This pops up a selection menu that lists all of the available data types. Select a data type such as **"delay"** from the list and click on the **"Apply"** button. Multiple data types can be compared at once by holding down the **"Shift"** key and selecting the data types followed by **"Apply"**. The units for each of the values displayed are 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

Figure 9: Data Type Selection Menu

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.

Holding down the left mouse button and dragging the rectangle over the plot area will zoom into the selected area (when the cursor is in the plot window). A single click of the right mouse key will zoom out 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

Liberate Details

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

IBIS Models

- Power Clamp & Ground Clamp Current
- Pull_up & Pull_down V-I Tables
- Rising & Falling Waveforms

Electromigration Models

- Electromigration Maximum Toggle Rate

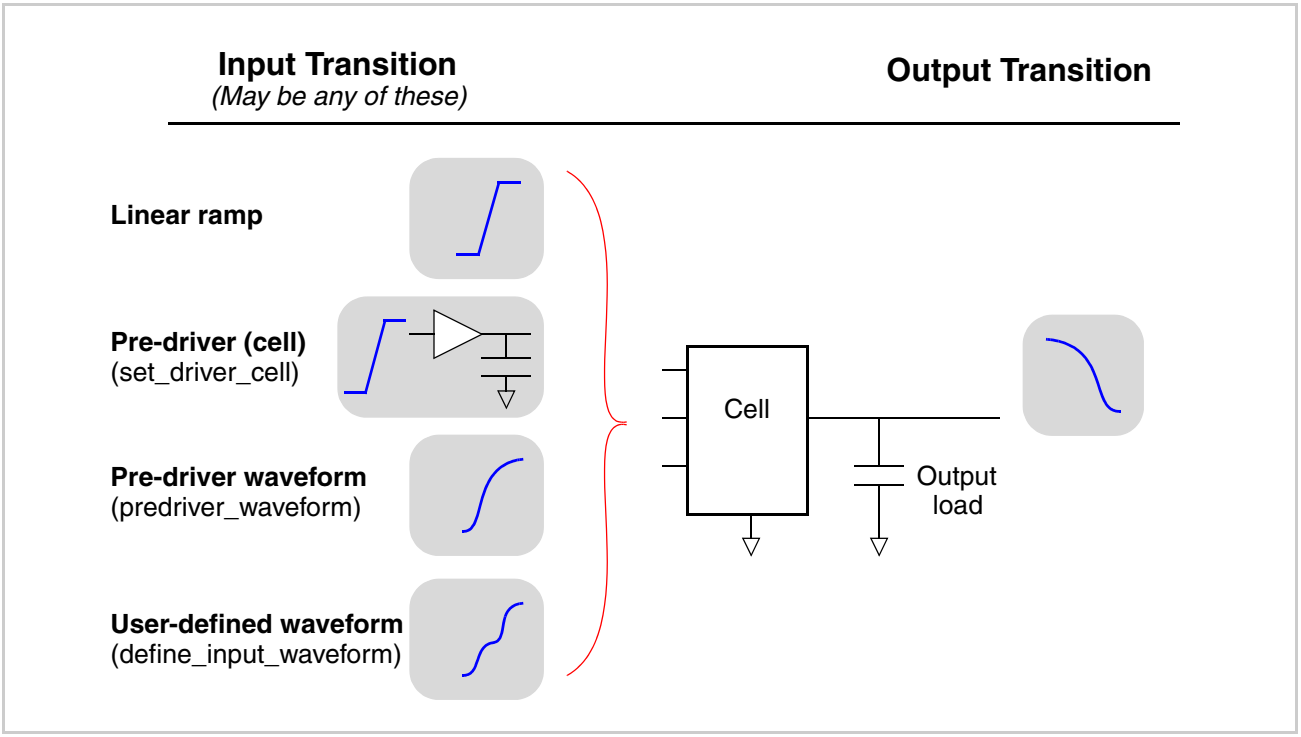
Delay Models

NLDM

The 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. The input transition can be any of these:

- linear ramp (from the defined template)
- output of a pre-driver (`set_driver_cell`)
- analytical driver waveform (`predriver_waveform`)
- user-defined waveform created using the `define_input_waveform` command.

The input pin is triggered by either 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-

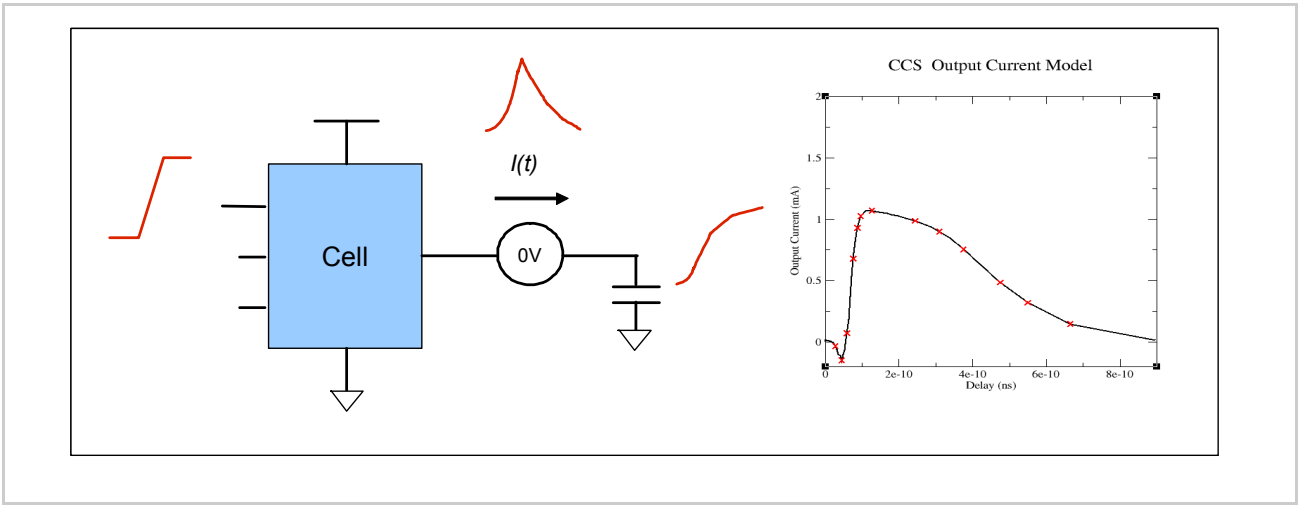
upper slew measurement points (default 30% to 70% of supply) for rising transitions and from the upper-to-lower measurement points (default 70% to 30% of supply) for falling transitions. The slew measurement points can be reset using the **measure_slew_***.

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 indices may be specified uniquely for each arc, cell or group of cells using the **define_index** and **define_template** commands.

CCS

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.

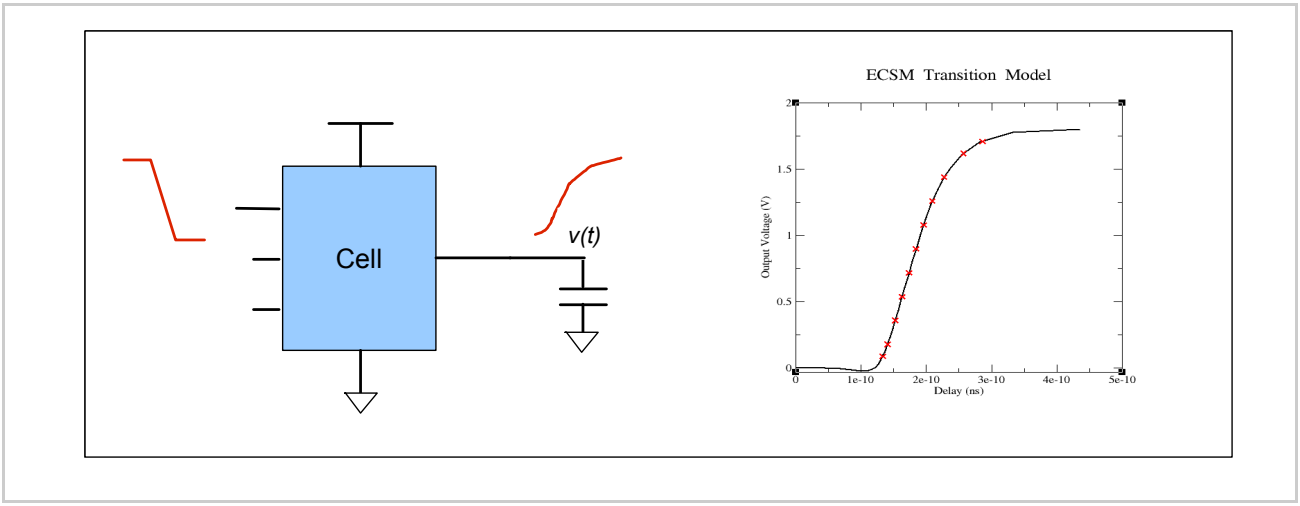
Dynamic selection of points ensures accuracy with minimal data size. The current waveform is measured such that the final voltage reaches within 1% of Vdd or Ground, and integrated

Voltage is calculated within 2%/2ps 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 that increasing the number of points and lowering the **ccs_abs_tol/rel_tol** will increase the library size. It has no impact on characterization runtime.

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.

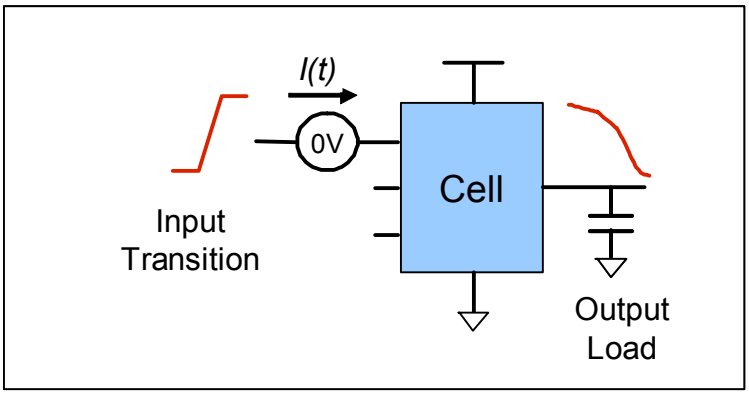


Circuit used for ECSM Driver Model

Pin Capacitance

NLDM Capacitance

Pin capacitance is characterized by measuring the current injected into the input pin over a time period, i.e. 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).

CCS Receiver Capacitance

The CCS receiver capacitance measurement utilizes the same circuit as the NLDM pin capacitance measurement, where current flowing into the input pin is captured during the input transition.

For rising transitions, current is measured from the input at 0 Volts to the delay measurement point (default 50% of supply) for the receiver_capacitance1, (C1) and from the delay measurement point (default 50% of supply) to the slew upper measurement threshold (default 70% of supply) for receiver_capacitance2 (C2). For falling transitions, current is measured

from the input transitioning from VDD to the delay measurement point (default 50% of supply) for the receiver_capacitance1 (C1), and from the delay measurement point (default 50% of supply) to the slew lower measurement threshold (default 30% of supply) for receiver_capacitance2 (C2). The actual capacitance values are determined by integrating the current over the range of the curve and dividing that by the change in voltage.

ECSM Capacitance

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 variable **constraint_glitch_hold**.

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.

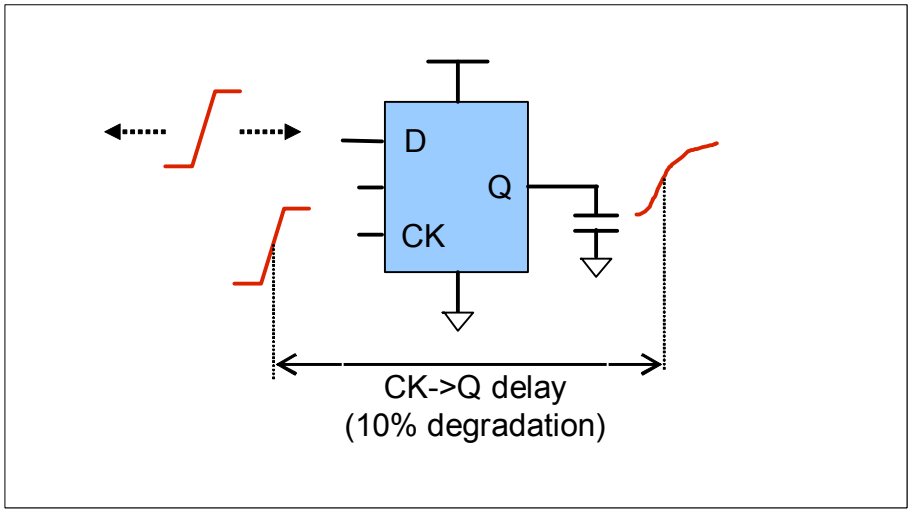
If **constraint_combinational**=1 and the pass/fail bisection fails, then use a delay based linear search for constraint characterization. In this method, a linear search is performed using a step size specified by the control variable **constraint_combinational_step_size** where a pass and a fail are 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:

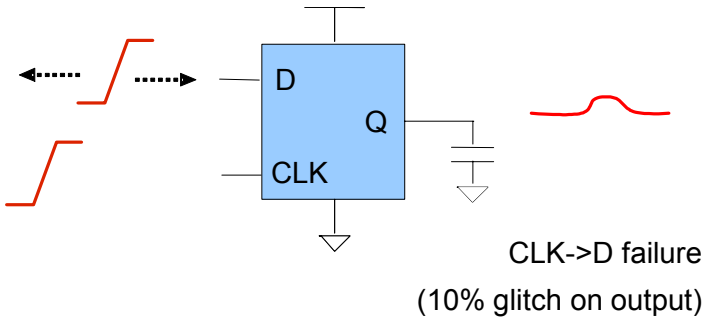
setup_rising rise_constraint value =
related_pin_transition - constrained_pin_transition + ${constraint_margin}
if < 0 then output "0" into the library
hold_falling fall_constraint value =
constrained_pin_transition - related_pin_transition + ${constraint_margin}
if < 0 then output "0" into the library

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 variable 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.



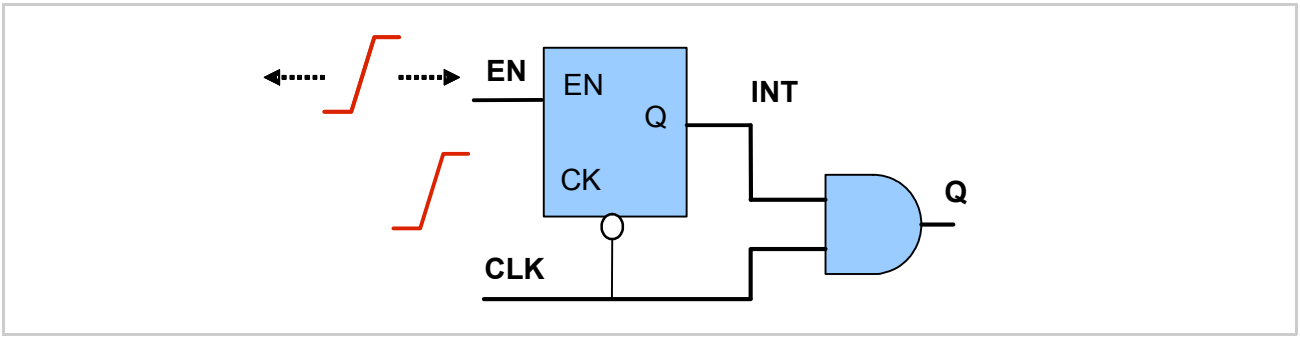
Setup and Hold Calculation



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** variable. 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".

Dependant constraint characterization

Liberate can characterize dependant setup and hold. An existing ldb can be used to re-characterize the dependent setup (or hold) time. Use the **constraint_dependent_setuphold** variable to enable dependent setup and hold characterization. A value of **1** requests re-characterization of setup. A value of **2** requests re-characterization of hold. To re-characterize for dependent setup and hold, an ldb must be loaded using read_ldb that includes standard setup and hold data. Default = 0 (standard setup and hold characterization). Use the

constraint_dependent_setuphold_margin variable to add a margin to the hold (setup) time when re-characterizing for dependent setup (hold) constraints. Use the **constraint_dependent_setuphold_margin_ratio** variable to add a margin to the hold (setup) time when re-characterizing for dependent setup (hold) constraints.

Usage:

```
read_ldb existing.ldb.gz# from a previous run
# Specify which constraint. Default: 0, 1=rechar setup, 2=rechar hold
set_var constraint_dependent_setuphold 2
char_library
write_ldb dependent.ldb
```

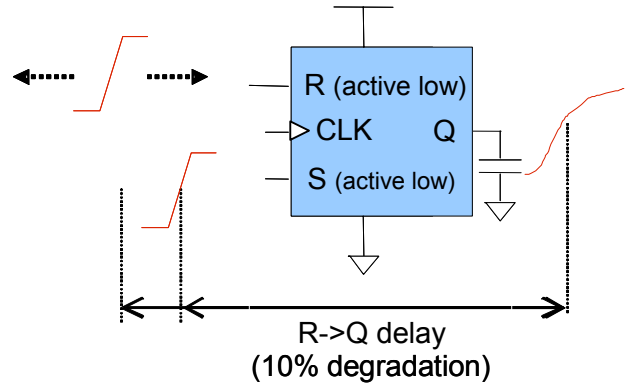
Recovery and Removal

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.**

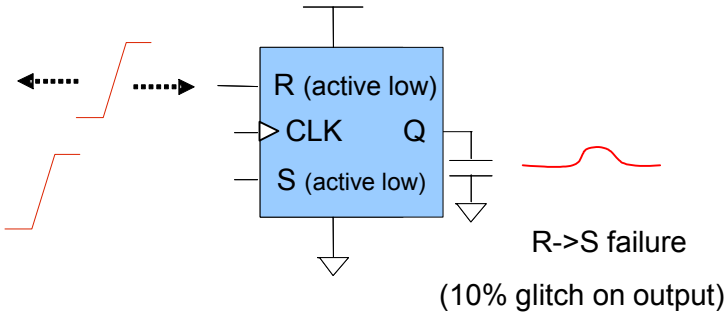
Non-sequential Setup and Hold

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 variable. 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 variable. 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 variable. In both cases, the final output voltage is checked that it is at the correct voltage. The **constraint_output_pin** variable 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 variable 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.

Power Models

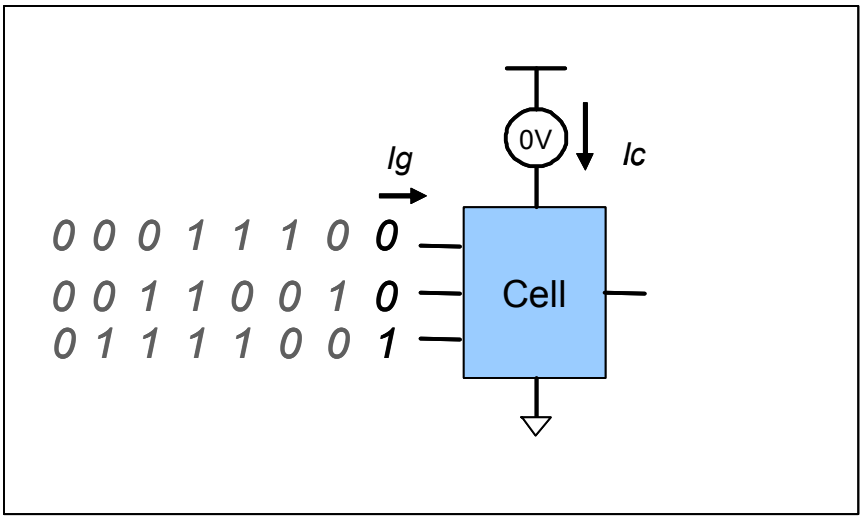
Power is modeled in a Liberty file in separate sections. Power is divided into leakage, hidden (internal_power under the input pin), and power (internal_power for switching outputs that is found under an output or bidi pin). The following sections will discuss how Liberate models these power types and the variables 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 / set_vdd 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** variable 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 variable.

The power_subtract_leakage variable can be used to enable the subtraction of the leakage power from the internal and hidden power. This variable 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** command for the other modes that are available.

The pin_based_leakage variable 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 variable 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 variable 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 variable 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 variable leakage_add_input_pin 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 variable max_leakage_vector to the desired maximum number of leakage states to characterize. If the default_leakage command is specified, then the *Liberate Inside View* will be disabled for leakage characterization, and only the user specified leakage *when* states will be characterized.

The conditional_leakage variable can be used to disable the output of state dependent leakage groups.

The keep_dcap_leakage variable 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 shortcircuit 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 variable conditional_hidden_power. When this variable 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 variable. 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 variable voltage_map (default=0) and the variable pin_based_power (default=1) to request that power be reported by power pin. With these variables 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 variable 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 Liberate's Inside View 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_map                    0      # (default)
set_var pin_based_power                2
set_var power_subtract_leakage         1      # (default)
set_var power_subtract_output_load     all    # (default)
set_var leakage_add_input_pin          1      # (default)
set_var leakage_mode                   1      # (default)
```

PG-pin syntax

1/2 of Vdd, Gnd, input energies, subtract 1/2 of output load energy

```
set_var voltage_map                    1
set_var pin_based_power                2
set_var power_subtract_leakage         1       # (default)
set_var power_subtract_output_load     all     # (default)
set_var leakage_add_input_pin          1       # (default)
set_var leakage_mode                   1       # (default)
```

No PG-pin syntax

Vdd energy, subtract output load energy on rise

```
set_var voltage_map                    0       # (default)
set_var pin_based_power                0
set_var power_subtract_leakage         1       # (default)
set_var power_subtract_output_load     all     # (default)
set_var leakage_add_input_pin          1       # (default)
set_var leakage_mode                   1       # (default)
PG-pin syntax
```

Vdd energy, subtract output load energy on rise

```
set_var voltage_map                    1
set_var pin_based_power                0
set_var power_subtract_leakage         0       # (default)
set_var power_subtract_output_load     all     # (default)
set_var leakage_add_input_pin          1       # (default)
set_var leakage_mode                   1       # (default)
```

Signal Integrity Models

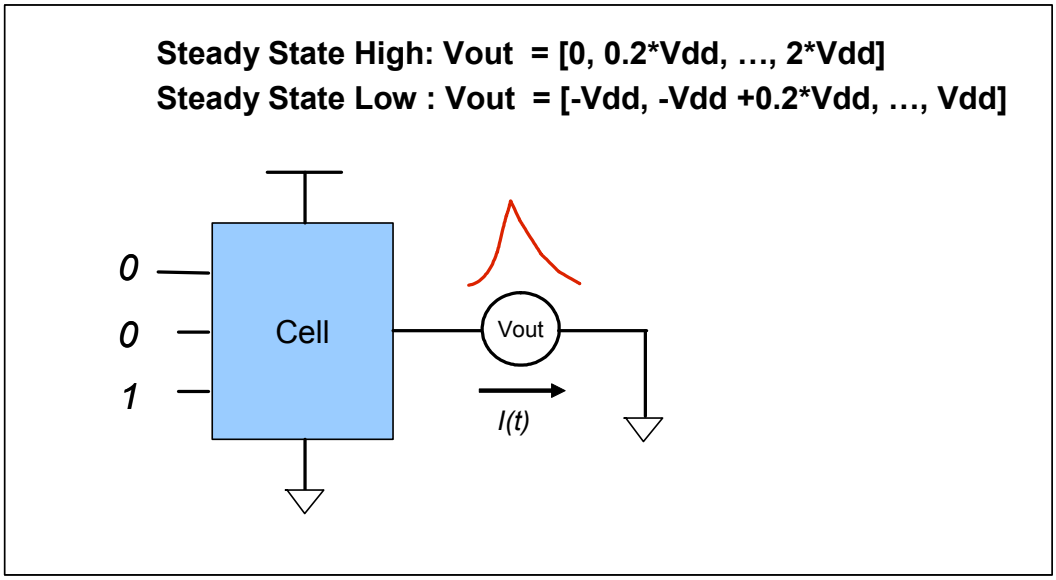
Liberate supports both Liberty SI models and composite current-source noise models (CCSN). The Liberty SI model contains constructs for steady state current, DC noise margins, hyperbolic noise curves and noise immunity tables. To characterize Liberty SI models, use the **si** option to **char_library**.

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-

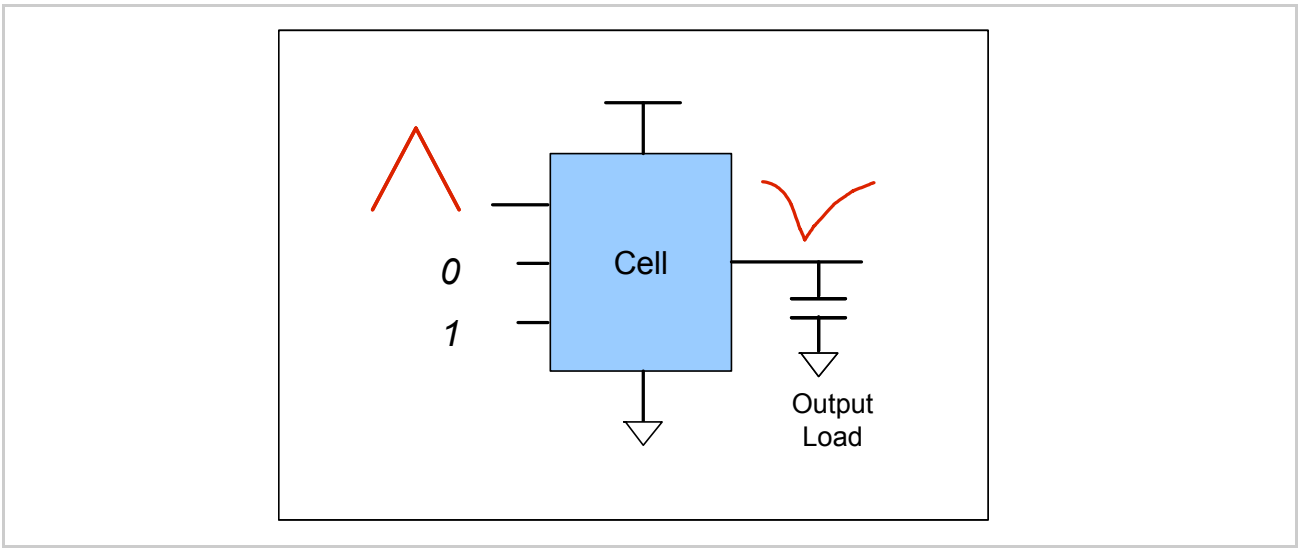
state low. The current through the voltage source is captured in the steady-state current table. This is performed for each distinct logic state where the output goes low or high. The intermediate voltage values to be tabulated can be specified by defining a template of type `si_iv_curve`.



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

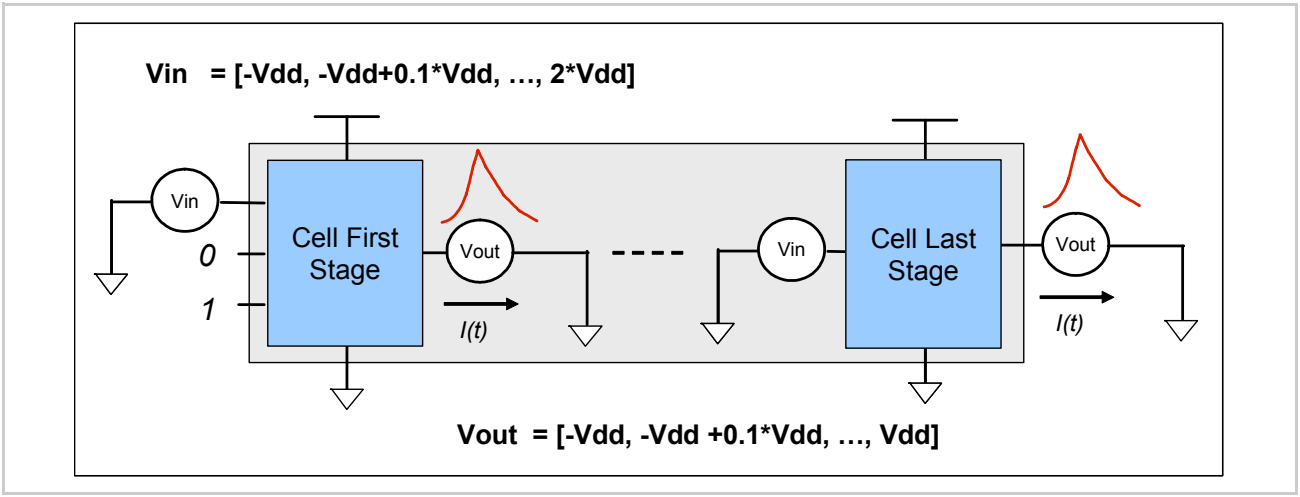
In addition to the noise-immunity curve, hyperbolic input noise (low and high) and DC noise-margins can be generated in the output library by setting the parameter **input_noise**. These values are extracted from the data captured by calculating the noise immunity curve. The hyperbolic area, height and 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.

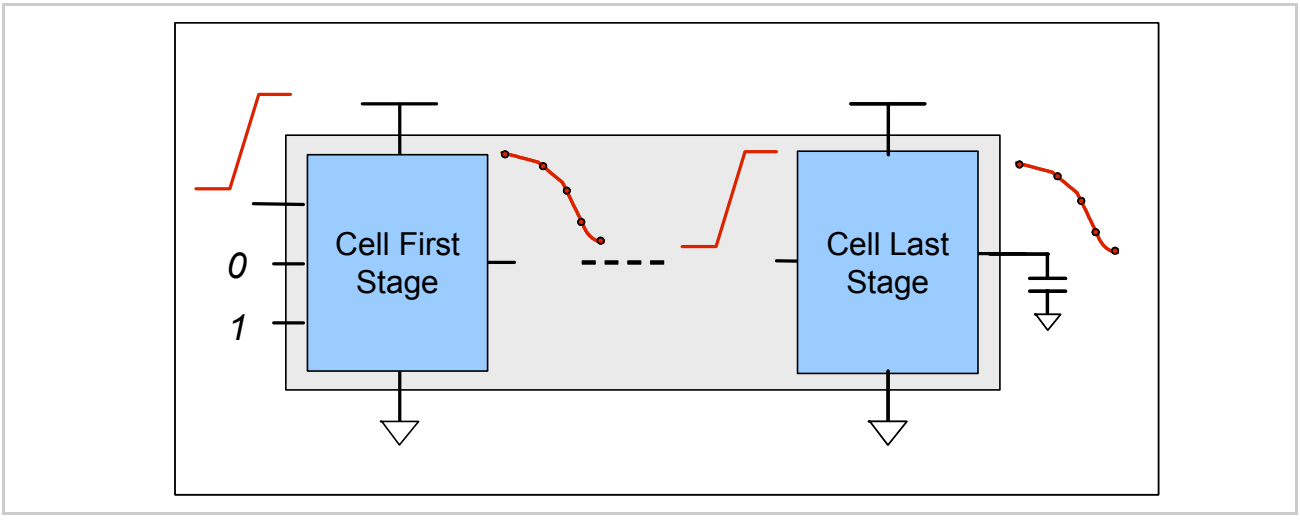
The voltage steps between –Vdd and 2*Vdd are determined by a pre-defined formula, such that 29 voltage points are sampled for each input and output voltage resulting in a 29X29 DC current table.



Circuit used for CCSN DC Current

CCSN Output Voltage

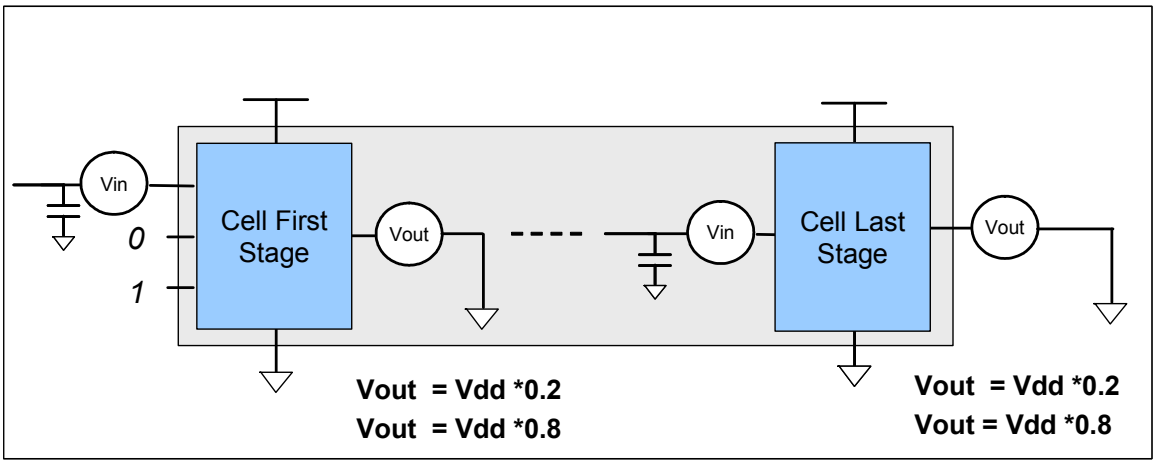
The CCSN output voltage model is calculated by simulating each stage using two linear slews from the input slew indices, and either two loads from the output load indices (if the stage output is a pin) or zero additional load capacitance (if the stage output is an internal node). Consequently, each stage 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

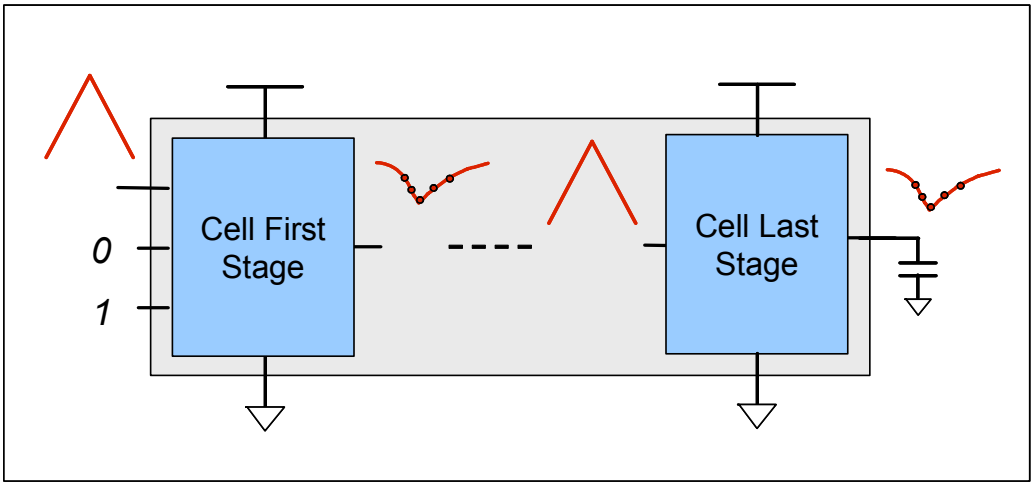
The CCSN Miller capacitance is calculated by placing a voltage source at the output of each stage and a capacitive load at the input pin of the stage. The Miller capacitance is measured by comparing the change in input voltage against a 20% change in output voltage for two different input loads.



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

IBIS Models

IBIS model content

Liberate supports the generation of models for IBIS versions 3.2 and 4.2. The process flow for characterizing IBIS models is described below.

The input/output buffer information specification (IBIS) is an approved standard within the Electronic Industry Alliance (EIA), and is also known as ANSI/EIA-656. Specifications of the IBIS format may be found The IBIS Open Forum website: www.eda.org/ibis

IBIS is a behavioral-modeling specification. It is a standard for describing the analog behavior of the buffers of a digital device using plain ASCII-text formatted data files. IBIS model files are used to perform signal integrity (SI) simulations of printed circuit boards. The information needed to perform this simulation are the voltage-current (V-I) characteristics and switching (output voltage versus time) characteristics of the buffer.

Each IBIS model file contains a general title section and six model sections containing behavioral model data for each pin:

1. Power_clamp

   Table of DC current values through the diode connected to Vcc rail, when a DC Voltage is applied to the output pin.

2. GND_clamp

   Table of DC current values through the diode connected to Gnd rail, when a DC Voltage is applied to the output pin.

3. Pull_up

   The V-I characteristics of the pullup circuit. This data is referenced to the pullup reference voltage: V_{table} = [Pullup Reference] – V_{in}, where V_{in} is the voltage referenced to ground.

4. Pull_down

   The V-I characteristics of the pulldown circuit. This data is referenced to the pulldown reference voltage: V_{table} = V_{in} – [pulldown reference], where V_{in} is the voltage referenced to ground

5. Rising Waveform

   a. With resistive load connected to pulldown reference

   b. With resistive load connected to pullup reference

6. Falling Waveform

   c. With resistive load connected to pullup reference

   d. With resistive load connected to pulldown reference

The Ramp value in the IBIS file is always taken from 20% - 80% and will overwrite the following parameters:

```
measure_slew_lower_rise
measure_slew_upper_rise
measure_slew_lower_fall
measure_slew_upper_fall
slew_lower_rise
slew_upper_rise
slew_lower_fall
slew_upper_fall
```

The Liberate IBIS modeling flow has been validated using examples of output, input, and tristate buffer cells. The current version of Liberate does not support the following IBIS constructs. These may be supported in future releases:

- Define Package Model
- '.pkg FILE' and '.ebd FILE'
- Series Pin Mapping
- Dynamic_clamp

Note: the content of IBIS files is case sensitive, except for reserved words and keywords.

Valid scaling factors allowed for IBIS are:

Symbol	Meaning	Power of 10	Decimal
t	tera	10^{12}	1,000,000,000,000.
g	giga	10^{9}	1,000,000,000.
M	Mega	10^{6}	1,000,000.
k	kilo	10^{3}	1,000.
m	milli	10^{-3}	.001
u	micro	10^{-6}	.000001
n	nano	10^{-9}	.000000001
p	pico	10^{-12}	.000000000001
f	fempto	10^{-15}	.000000000000001

Base units are: Volts, Amperes, Ohms, Farads, Henries, Celsius, and Seconds.

Full abbreviations for the units are allowed, (e.g., pF, nH, mA, mOhm). In addition, scientific notation is supported (e.g., 1.2345e-12)

IBIS modeling flow

Liberate supports the generation of IBIS models using a flow similar to the Liberate Liberty library generation. An IBIS template must be written for each PVT corner IBIS-specific definition. Each PVT is then characterized using the Liberate command "**char_library -ibis -io**". A separate Liberate database file (ldb) is created for each. These characterized results are then merged into a single IBIS file using the Liberate command "**write_ibis_file $file $ldbs $cells".**

The first step is to run liberate to generate one to three set of IBIS enabled LDBs to cover typ, min and max corners. The min corner refers to the simulation setting with minimum voltage. The max corner refers to the simulation setting with maximum voltage. If only one ldb is used, then it is assigned as the 'typ' corner. If two set of LDBs are used, then they are 'typ' and 'min' corners.

The following are required to enable IBIS mode in ldb.

1. Describe the IBIS structure to Liberate:

(required)

```
ibis_define_header
```

```
ibis_define_component
ibis_define_model
ibis_define_pin
```

(depends on IBIS model type)

```
ibis_define_waveform_template
ibis_define_model_selector
```

2. Provide define_arc commands to instruct Liberate to generate I/V and V/t simulation data for IBIS. In the char_library command, specify the **io** and **ibis** flags to run Liberate in IBIS mode using user defined arcs.

Example:

```
set_operating_condition -voltage 3.3 -temp 25
source template_arcs.tcl
source template_ibis.tcl
char_library -ibis -io -thread 2 -extsim hspice
write_ldb MyCell_typ.ldb
```

After all corner LDBs are generated, run Tcl command 'write_ibis_file' from Liberate to write out IBIS file.

Example :

```
set cells "MyCell"
set file "MyCell.ibs"
set ldbs { MyCell_typ.ldb.gz MyCell_min.ldb.gz MyCell_max.ldb.gz }
write_ibis_file $file $ldbs $cells
```

A three corner IBIS model flow will look like this:

```
              char library                      write ibis file
{typ PVT IBIS.tcl} -> Liberate -> [typ.ldb] \
{min PVT IBIS.tcl} -> Liberate -> [min.ldb]  |--> Liberate --> [cell.ibs]
{max PVT IBIS.tcl} -> Liberate -> [max.ldb] /
```

IBIS-specific templates must be prepared for each cell. These are defined using these IBIS-specific commands: ibis_define_header, ibis_define_waveform_template, ibis_define_component, ibis_define_model, ibis_define_pin, ibis_define_model_selector. Each of these IBIS-specific commands is described in detail in the "Liberate Commands" chapter.

Example: Liberate command sequence for an IBIS cell characterization:

```
# characterize 3 corners, typ, min, max
set_operating_condition -voltage 3.3 -temp 25

# defines all of the measurement arcs
source template_arcs.tcl

# defines headers, waveform templates, components, pins,
# and models
source template_ibis.tcl
```

```
# characterize using IBIS mode and save the database
char_library -ibis -io
write_ldb lvds_typ.ldb

# After all corners are characterized, write the
# IBIS cell model
set file "OBuff.ibs"
set ldbs { OBuff_typ.ldb.gz \
OBuff_min.ldb.gz \
OBuff_max.ldb.gz }
set cells "MyOBuff"
set versions {4.3 3.2}
set revision "1.1"
write_ibis_file -version $versions -rev $revision \
$file $ldbs $cells


###*******************************************************
### Liberate template example for Tri-state IO cell ******
###*******************************************************

## Measure [Ramp] using 20% and 80% threshold
set_var slew_lower_rise 0.20
set_var slew_lower_fall 0.20
set_var slew_upper_rise 0.80
set_var slew_upper_fall 0.80

set_var measure_slew_lower_rise 0.20
set_var measure_slew_lower_fall 0.20
set_var measure_slew_upper_rise 0.80
set_var measure_slew_upper_fall 0.80

## Define input slew (1ns, and no loading cap)
define_template -type delay \
         -index_1 {1.0 } \
         -index_2 {0.0 } \
         delay_template_1x1

define_template -type power \
         -index_1 {1.0 } \
         -index_2 {0.0 } \
         power_template_1x1

## Define cell
define_cell \
       -input { A EN PUPD } \
       -output { Y } \
       -bidi { PAD } \
       -pinlist { A EN PUPD PAD Y } \
       -delay delay_template_1x1 \
       -power power_template_1x1 \
       MyCell

## Define pin pull_up and pull_down (must be consistent
## with R_load and V_fixture from IBIS)
define_pin_load \
         -pullup_voltage 3.3 \
         -pullup_resistance 50 \
         load_template_up

define_pin_load \
```

```
          -pulldown_resistance 50 \
          load_template_down

# These define_arc commands generate I/V and V/t
# waveforms for model when condition '(!PUPD)'.

# Si-IV steady_state_current_low : PAD pull_down
define_arc \
       -vector {R00RX} \
       -when "!PUPD" \
       -related_pin A \
       -pin PAD \
       MyCell

# Si-IV steady_state_current_high : PAD pull_up
define_arc \
       -vector {F00FX} \
       -when "!PUPD" \
       -related_pin A \
       -pin PAD \
       MyCell

# These 4 define_arc commands generate rising/falling
# waveform V/t table

# PAD Rising Waveform Pullup On
define_arc \
       -vector {R00RX} \
       -when "!PUPD" \
       -pin_load load_template_up \
       -load_dir U \
       -related_pin A \
       -pin PAD \
       MyCell

# PAD Rising Waveform Pulldown Off
define_arc \
       -vector {R00RX} \
       -when "!PUPD" \
       -pin_load load_template_down \
       -load_dir D \
       -related_pin A \
       -pin PAD \
       MyCell

# PAD Falling Waveform Pulldown On
define_arc \
       -vector {F00FX} \
       -when "!PUPD" \
       -pin_load load_template_down \
       -load_dir D \
       -related_pin A \
       -pin PAD \
       MyCell

# PAD Falling Waveform Pullup Off
define_arc \
       -vector {F00FX} \
       -when "!PUPD" \
       -pin_load load_template_up \
       -load_dir U \
```

```
        -related_pin A \
        -pin PAD \
        MyCell

# These 2 define_arc commands generate waveform for
# Power_clamp and GND_clamp I/V table

# Tri-state Enable, steady_state_current_low :
# Power_clamp/GND_clamp
define_arc \
        -type enable \
        -vector {1F0RX} \
        -related_pin EN \
        -pin PAD \
        MyCell

# Tri-state Enable, steady_state_current_high :
# Power_clamp/GND_clamp
define_arc \
        -type enable \
        -vector {0F0FX} \
        -related_pin EN \
        -pin PAD \
        MyCell

# ******************************************************
# The following are define_arc commands to model an OBuff
# output pad. A subckt file: 'termres.sp' is included
# during characterization. A custom output load (harness)
# defined in termres.sp, is applied to OBuff to model
# PAD and PADN pin loading. The 'mid' voltage is taken
# from a previous measurement of the cross voltage of PAD
# and PADN signals with a 100ohm resistor connected as
# termination. 1.2V is used here as typ value.
# For standard timing characterization (delay, transition),
# a global pin cap 'PAD_cap' and 'PADN_cap' is defined in
# the top level, which must be connected here.
# ******************************************************

## an example of termres.sp file
.subckt termres_typ pad padn
R1 pad  mid 50
R2 padn mid 50
Vmid mid 0 1.20
.ends
.subckt termres_min pad padn
R1 pad  mid 50
R2 padn mid 50
Vmid mid 0 1.195
.ends
.subckt termres_max pad padn
R1 pad  mid 50
R2 padn mid 50
Vmid mid 0 1.205
.ends
.subckt termres pad padn
R1 pad padn 100
C1 padn 0 'PADN_cap'
C2 pad  0 'PAD_cap'
.ends
```

```
# The define_cell command includes this harness cell
define_cell \
    -input {A SEL PWRDN} \
    -output {PAD PADN} \
    -pinlist {A SEL PWRDN PAD PADN} \
    -harness "termres_typ" \
    -delay delay_template_OL_1x1 \
    -power power_template_OL_1x1 \
    MyOBuff

# A set of define_arc commands for generating
# pullup/pulldown I/V table

# Si-IV steady_state_current_high : PAD pull_up
define_arc \
    -vector "F00FR" \
    -when "!SEL" \
    -related_pin A \
    -dual_pin PADN \
    -pin PAD \
    MyOBuff

# Si-IV steady_state_current_low : PAD pull_down
# ECSM : PAD Rising Waveform Differential
define_arc \
    -vector "R00RF" \
    -when "!SEL" \
    -dual_pin PADN \
    -related_pin A \
    -pin PAD \
    MyOBuff
```

Example:

The Tcl command for Liberate IBIS :

```
###********************************************
# The following commands can be used in the header:
#
#    [IBIS Ver]  version             <-- required
#    [Comment Char] comment token    <-- optional
#    [File rev]  rev                 <-- optional
#    [Source]    source              <-- optional
#    [Notes]     notes               <-- optional
#    [Disclaimer]    disclaimer      <-- optional
#    [Copyright] copyright           <-- optional

## ibis_define_header {}
  -ibis_ver         : IBIS version, default '4.2'
  -file_rev         : user IBIS file version
  -comment_char     : IBIS comment character, default "|"
  -source           : User specified source description
  -notes            : User specified note
  -disclaimer       : User specified disclaimer text
  -copyright        : User specified copyright text
  filename          : IBIS output file name


## ibis_define_component {}
  -manufacture      : Component's manufacturer name
```

```
  -package          : a list of package R/L/C at typ/min/max condition
  -waveforms        : a list of pullup/pulldown waveform templates
  -net_typ_vdd      : a list of {net typ_vdd} pairs
  -end_time        : The maximum waveform end time to write to IBIS rise/fall
table
  components        : IBIS component name(s) list

## This command defines waveforms for use in the ibis_define_component command
## ibis_define_waveform_template {}
  -type             : 'rise' or 'fall'
  -vfix             : V_fixture
  -rfix             : R_fixture
  -lfix             : L_fixture
  -cfix             : C_fixture
  -pins             : list of pins to apply this waveform
  name              : Waveform template name

## ibis_define_pin {}
  -net              : Pin net/signal name from Spice netlist
  -model            : Pin model name
  -enable           : Enable pin signal (port) name from Spice netlist
  -function         : Logic function to set pin to logic 1 (high) state (the
when logic)
  -inv_pin          : The corresponding inverting pin name for I/O output
  -inv_pin_net      : The inverting pin signal (port) name
  -vdiff            : The differential receiver threshold voltage between pin
and inv_pin
  -tdelay_typ       : The typ launch delays of the inv_pin relative to the pin
  -tdelay_min       : The min launch delays of the inv_pin relative to the pin
  -tdelay_max       : The max launch delays of the inv_pin relative to the pin
  -R_pin            : Pin package R value
  -L_pin            : Pin package L value
  -C_pin            : Pin package C value
  -components      : IBIS component name(s) list which use this pin definition
  pin               : Pin name

## ibis_define_model {}
  -model_type     : Pin model type, supported type : Input, Output, I/O, 3-state
  -when             : When condition to match waveforms' when logic
  -polarity         : "Non-Inverting" or "Inverting"
  -enable           : "Active-High" or "Active-Low"
  -vinl             : Maximum lower threshold voltage
  -vinh             : Minimum upper threshold voltage
  -vmeas            : Reference voltage for timing measurements
  -cref             : Timing specification test load capacitance
  -rref             : Timing specification test load resistance
  -vref             : Timing specification test load voltage
  -cref_diff        : Timing specification differential capacitance
  -rref_diff        : Timing specification differential resistance
  -cfix_f           : Falling waveform C_fixture
  -r_load           : Loading resistor for [Ramp] measurement
  -component        : this model is defined for these component(s) list
  modelName         : IBIS model name

## ibis_define_model_selector {}
  -model_desc       : list of model and description in pair(s)
  -component        : this model is defined for these component(s) list
  msName            : model selector name
```

Example :

```
    set cells {MyLVDS}

    set mslist { \
    LOW_VOLT_PUPD    "Low Voltage Pull_up enabled"   \
    LOW_VOLT_!PUPD   "Low Voltage Pull_down enabled" \
    }

    ibis_define_model_selector \
    -model_desc $mslist \
    -component $cells \
    LVDS_OUT


## write_ibis_file {}
  file     : IBIS file to write to
  ldbs     : List of ldb file for typ/min/max corners
  cells    : List of cell names
```

For example:

```
set cells "MyCell"
set file "Cell.ibs"
set ldbs { MyCell_typ.ldb.gz MyCell_min.ldb.gz \ MyCell_max.ldb.gz }
write_ibis_file $file $ldbs $cells
```

Truth Table Example for IBIS

The Truth Table based IBIS flow contains the following steps :

A) Use an existing truth_table template or start from scratch to compose a truth table file with IBIS specific commands to describe the cell pin behavior and IBIS information. This file should contain at least the following structure:

Example:

```
* LIB=MyLib
* CELL=buffer
* TABLE= A OEN PAD @ PAD C
         0  0   -  @  0  -
         1  0   -  @  1  -
         -  1   0  @  -  0
         -  1   1  @  -  1
         -  1   Z  @  -  X
* TABLE_END
* CELL_END
#
* IBIS_BEGIN
* IBIS_VER=4.2
* IBIS_FILE_REV=0.1
* IBIS_FILE=buffer.ibs
* CORE_TEMPERATURE={25,125,-40}
* CORE_VOLTAGE={1.1,0.99,1.32}
....
....
* IBIS_END
* CELL_END
```

B) Create a script to generate template files.

Example:

```
% cat > gen.tcl
    read_truth_table buffer.tt
    write_template -truth_table -ibis -ibis_slew 0.1 template_arc.tcl
```

A) Run Liberate

```
% liberate gen.tcl
```

B) Review the message generated by Liberate for any warning and error.

If there is no error and if all three corner data are provided, then you can find the following files been generated:

char_all.csh	Shell script file to run all char works and generate IBIS file
common.tcl	An empty file for users to define simulator and model/netlist path and common variable settings
ibis_buffer_max.tcl	Char script for max voltage corner
ibis_buffer_min.tcl	Char script for min voltage corner
ibis_buffer_typ.tcl	Char script for typ voltage corner
igen_buffer.tcl	Script to read in ldb.gz file and generate IBIS file
predef.tcl	An empty file for users to define variables used in template_arc.tcl and template_ibis.tcl.
postdef.tcl	An empty file for users to define variables used in **char_library** section
template_arc.tcl	Template file contains define_cell{} and define_arc{} commands
template_ibis.tcl	Template file contains IBIS related define_ibis command.

If the files **common.tcl**, **predef.tcl**, and **postdef.tcl** exist they will not be overwritten.

Sample run script showing loading sequence (ibis_buffer_typ.tcl):

```
...
set cells {}
set corner typ

source predef.tcl

source ${run_dir}/template_arc.tcl
source ${run_dir}/template_ibis.tcl

source common.tcl

read_spice $modelfiles
read_spice $spicefiles

source postdef.tcl

char_library
```

Electromigration Models

Electromigration Model Content

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" 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 Modeling Flow

Electromigration Characterization Flow Overview

This Electro-Migration (EM) 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.

Summary of EM Characterization and Modeling Flow

Creation of an EM liberty model in Liberate can be accomplished in the following five steps:

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 Spectre version MMSIM13.1 ISR12 or later.

4. Generate a liberty library with EM data by adding the command line argument `-em` to the **write_library** command.
5. 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 variables 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 variables that can be used to tune the EM characterization. For more information, see the variables prefixed by "em" in the Liberate Parameters chapter of the Liberate manual.

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 indices 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 indices 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 variable max_capacitance_auto_mode.

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 variable auto_index_distinct_risefall to have Liberate compute a separate max_capacitance for rise and fall transitions.

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...}

This is used to specify 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. Example:

```
* ASYNC_PINS={A B C}
```

BIDI_PINS={Pin1 Pin2...}

This is used to specify 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
```

So the pin PAD will be added to the output option of the define_cell command:

```
define_cell -output {PAD} ...
```

CELL=<cell_name>

Specify 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...}

This is used to specify 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>

Specify the constraint template name to be inserted to define_cell.

DELAY_TEM=<name>

Specify 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.)

This is needed for IBIS template generation for dual input/output model type cells such as LVDS cells. When this argument is present, Liberate adds **-dual_pin** and **-dual_dir** options in the define_arc command in the generated template file.

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[,..]...

This can be used to specify 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 ...] >

This is used to tell 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...}

This is used to specify 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, ...}

This is used to define 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. Example:

```
* LEAKAGE_ENUM_PINS={PAD data_out data_out_en st_en}
```

LIB=<library_name>

This can be used to specify the library name. Currently this is only used when generating IBIS format output. The library name specified is used as the ID for .ldb file. Example:

In buffer.tt we have:

```
* LIB=MyLib
```

Then the final IBIS ldb files would be:

```
ibis_MyLib_typ.ldb.gz
ibis_MyLib_min.ldb.gz
ibis_MyLib_max.ldb.gz
```

These names will be used in igen_<cellname>.tcl file as well.

MAP_IGNORED_BIT=<character>

This can be used to map 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...}

This is used to specify 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

Specify 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,...>

This is used to identify differential pin pairs. Multiple pin pairs can be specified when separated by a comma (","). Note: This command is not supported at this time and will be supported in a future release.

POWER_TEM=name

Specify the power template name to be inserted to define_cell

PULLUP=<pins>
PULLDOWN=<pins>

These commands are used to specify 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>

These commands specify 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>

Used to specify the pull-up voltage (float, in Volts)

RELATED_PINS={Pin:Related_Pin [,Related_Pin ...] }

This command is to tell 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>

Used to specify the series connected resistance (float, in Ohms) to add to the bi-directional pin.

SI_IM_TEM=name

Specify the SI Immunity template name to be inserted to define_cell

TABLE=Inputs@Outputs
<state_values>
TABLE_END

Use this command to specify the actual 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>

Use this to specify the default table separator value.

(Default: '@'). Specify this command at the very beginning of the input file (not inside any *CELL) to specify separator globally.

WRITE_HIDDEN_POWER = < 0 | 1 >

Use this option to control 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 >

Use this option to add the '**-when**' option to the **define_arc** commands. When in IBIS mode, the '-when' is always added. 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
```

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 indices.

```
read_truth_table io_cells.table
write_template -auto_index \
    -index_delay 5x6 \
    -truth_table template_io_cells
```

Truth Table Format – IBIS

Notes on command syntax

Some data lines might be too long to fit in one line. In this case, quotes can be used to continue the command onto multiple lines. For example:

```
* DISCLAIMER="This software contains information
    confidential and proprietary
    to ABC Corporation."
```

Use quotes if there are spaces in the data:

```
* MANUFACTURER="Cadence Design Systems, Inc."
```

For data fields such as {25,-40,125}, ***don't put a space*** in between the values. *Example of incorrect formatting:* {25, -40, 125}.
A space (or ;) is to separate data fields such as {P1:PAD P2:COUT}.

A global command, like CORE_VOLTAGE, SOURCES, and IBIS_FILE can appear only once per table.

Some commands to set IBIS pin data may be utilized many times in a section.

Example:

```
* SET_POWER_NET_VDD={VDDS:1.8,1.62,1.98 BIASFET:1.8,1.62,1.98}
* SET_POWER_NET_VDD={BIAS1:1.25,1.25,1.25 BIAS2:1.25,1.25,1.25}

* MODEL_WHEN={Buffer_Output:!LOPWRA&!LOPWRB}
* MODEL_WHEN={Buffer_Output:LOPWRA&!LOPWRB }
* MODEL_WHEN={Buffer_Output:!LOPWRA&LOPWRB }
* MODEL_WHEN={Buffer_Output:LOPWRA&LOPWRB  }
```

For IBIS Truth Table commands, a PIN refers to the IBIS pin name and a NET refers to the Spice netlist net name.

For example, the user may have these PIN/NET/MODEL in the file:

```
# IBIS_PIN  PIN_NET MODEL
#  P1       PAD   Buffer_Output
#  P2       PTST  Buffer_TriState
#  PW1      VDDS  POWER
#  GW1      VSSS  GND
```

Liberate uses the following information to link them together:

```
* PIN_MODEL_NAME={P1:Buffer_Output}
* PIN_NET_NAME={P1:PAD}
* MODEL_TYPE={Buffer_Output:Output}
* PIN_MODEL_NAME={P2:Buffer_TriState}
* PIN_NET_NAME={P2:PTST}
* MODEL_TYPE={Buffer_TriState:3-state}
```

Example of dual related output pins (PAD and PADN):

```
* PIN_NET_NAME={P1:PAD}
* PIN_INV_PIN={P1:P2}
* PIN_INV_PIN_NET={P2:PADN}
```

IBIS Truth Table Commands

C_PKG={typ,min,max}
L_PKG={typ,min,max}
R_PKG={typ,min,max}

Define component pins default packaging values, C/L/R in Farads/Henrys/Ohms.

COPYRIGHT="<string>"

This specifies the copyright string to be included in the IBIS output file.

CORE_TEMPERATURE={typ[,min[,max]]}

This specifies the 3 PVT temperatures that the IBIS will be characterized at. The value is in a list: {typ min max}.

CORE_VOLTAGE={typ[,min[,max]]}

This specifies the 3 PVT voltages that the IBIS will be characterized at.

DISCLAIMER="<string>"

This specifies the disclaimer string to be included in the IBIS output file.

IBIS_BEGIN
IBIS_END

Specifies the beginning and ending of an IBIS section.

IBIS_FILE="<filename>"

This specifies the IBIS output file name. The file name should be in all lowercase. If not, Liberate will automatically convert it to lower case to fulfill the IBIS requirement. If the suffix is not ".ibs", Liberate will append ".ibs" to the end.

IBIS_FILE_REV="<revision>"

This specifies the output file revision.

IBIS_VER="<version>"
(or IBIS_VERSION)

This specifies the IBIS version in the output IBIS file. Default is "4.2". There is a limitation of up to 100 data points in some sections in a version 3.X file. This limit is changed to 1000 data points in version 4.X. (Also accepted: IBIS_VERSION)

MANUFACTURER="<string>"

This specifies a manufacturer string to be included in the IBIS output file.

MODEL_C_REF/MODEL_R_REF/MODEL_V_REF={model_name:pin_name ...}

Define the timing specification test load (Cref, Rref, Vref) in the [Model] section. Please see the IBIS specification for more details.

Example:

```
* R_PKG={0,0,0}
* PIN_R_PIN={Out:200m PAD:210m In:50m}
* MODEL_R_REF={Buffer_Output:50}
```

MODEL_ENABLE_PIN_NET={model_name:pin_name ...}

This command is used to clarify the Enable attribute of an IBIS [Model] section. If not provided, Liberate will attempt to determine the enable pin from the truth table, provided there is only one enable pin/net. Use this command to set the enable pin netname for a IBIS model. For example, assume the cell TriBuffer has an enable pin, a pad pin and one output pwrdn pin. The pin used to enable/disable the output pin PAD of the cell TriBuffer is OEN (OEN -> PAD). In order to help Liberate to choose the OEN pin (instead of PWRDN) is the enable pin for PAD, the user can set this command:

```
* MODEL_ENABLE_PIN_NET={TriBuffer:PAD}
```

MODEL_RELATED_PIN_NET={model_name:pin_name ...}

Specify the model pin to net mapping. Use this command to set the related pin netname for an IBIS model. For example, assume a cell Buffer_Output has multiple input pins and control pins, and the main input netname to drive the output DOUT of the model Buffer_Output is A (A -> DOUT). In order to tell Liberate that control pins LOPWA and LOPWB are not directly related pins for DOUT, use the following command:

```
* MODEL_RELATED_PIN_NET={Buffer_Output:A}
```

This command is also used to determine the Polarity attribute of an IBIS [Model] section. Liberate determines the polarity direction (Inverting / Non-Inverting) from the truth table and puts it into the IBIS model Polarity attribute.

MODEL_TYPE={model_name:type}

Specify the type for each model.

Fully supported model types:

```
Input
Output
I/O
3-state
Open_drain
Open_sink
Open_source
I/O_open_drain
I/O_open_sink
I/O_open_source
POWER
GND
```

Note: The truth table feature does *not* support **open_drain/open_sink** and **open_source** types. Users should use the standard (I/O, Output) model first, then manual modify the generated template_arc.tcl and template_ibis.tcl files to remove the pull up/rising arcs that are not needed.

Example:

```
* MODEL_TYPE={Buff_I:Input Buff_O:Output Buff_PAD:3-state}
```

MODEL_VINH={pin_name:Vinh ... }

Specify the input logic high DC voltage. The Vinh can accept one, two, or three entries with the voltage values mapping to: "typ", "min", and "max". The typical value is listed in [Model] section. If more than one voltage is specified, the typ/min/max values are listed in the [Model Spec] section when there is a difference between them.

Example 1:

```
* MODEL_VINH={Buff_PAD:2.0 Buff_In:2.0}
```

Example 2:

```
* MODEL_VINH={Buffer_In:2.31,2.08,2.54}
```

MODEL_VINL={pin_name:Vinl ... }

Specify the input logic low DC voltage. The Vinl can accept one, two, or three entries with the voltage values mapping to: "typ", "min", and "max". The typical value is listed in the [Model] section.

If more than one voltage is specified, the typ/min/max values are listed in the [Model Spec] section when there is a difference between them.

Example 1:

```
* MODEL_VINL={Buffer_PAD:0.8 Buffer_In:0.8}
```

Exmaple 2: input buffer :

```
* MODEL_VINL={Buffer_In:1,0.89,1.09}
* MODEL_VINH={Buffer_In:2.31,2.08,2.54}
```

Liberate command generated from truth table

```
ibis_define_model \
    -model_type Input \
    -vinl {1.0 0.89 1.09} \
    -vinh {2.31 2.08 2.54} \
    -when "(!PI&!PUPDSEL)" \
    -component $ibisCellName \
    P1_!PI_!PUPDSEL

( input buffer IBIS )
[Model]                    P1_!PI_!PUPDSEL
Model_type          Input
Vinl = 1.0
Vinh = 2.31

[Model Spec]
Vinl                        1.0           0.89          1.09
Vinh                        2.31          2.08          2.54
```

MODEL_VMEAS={pin_name:Vref ... }

Specify the reference voltage for timing measurements. The Vref can accept one, two, or three entries with the voltage values mapping to: "typ", "min", and "max". The typical value is listed in the [Model] section.

If more than one entry is given, the typ/min/max values are listed in the [Model Spec] section when there is a difference between them.

Example 1:

```
* MODEL_VMEAS={Buff_PAD:1.65 Buff_O:1.65}
```

Example 2: output buffer

```
* MODEL_VMEAS={Buffer_Out:1.21,1.195,1.215}
```

Liberate command, generated from truth table

```
ibis_define_model \
    -model_type Output \
    -polarity Non-Inverting \
    -r_load 50 \
    -vmeas {1.21 1.195 1.215} \
    -rref 101M \
    -cref 15pF \
    -vref 0.0 \
    -when "(!LPSEL&!LOPWRA&!LOPWRB)" \
    -component $ibisCellName \
    P1_!LPSEL_!LOPWRA_!LOPWRB

( output buffer IBIS )
[Model]                  P1_!LPSEL_!LOPWRA_!LOPWRB
Model_type          Output
Polarity    Non-Inverting
Vmeas = 1.21

[Model Spec]
Vmeas                     1.21        1.195     1.215
```

MODEL_WHEN={model_name:<resistance>,...}

Specify the *when* condition(s)used in each model. If multiple MODEL_WHEN= commands exists for a model, then Liberate will group them together as one [Model Selector], and create a name for each when condition. The user may change the name and add description of the generated template_arc.tcl file. Example:

```
* MODEL_WHEN={Buff_O:PU&PD,PU&!PD,!PU&PD,!PU&!PD}
```

NOTES="<string>"

This specifies a note string to be included in the IBIS output file.

PIN_C_FIXTURE={pin_name:capacitance}

Specify the C_fixture (test fixture) for an IBIS pin.

PIN_L_FIXTURE={pin_name:inductance}

Specify the L_fixture (test fixture) for an IBIS pin.

C_fixture and L_fixture can be used to produce waveforms which describe the typical test case setups for reference. However, they are generally not recommended.

(Note, there is no PIN_R_FIXTURE command since PULLUP_RES/PULLDN_RES is the R_Fixture.)

PIN_C_PIN = {pin_name:<capacitance> ...}
PIN_L_PIN = {pin_name:<inductance> ...}
PIN_R_PIN = {pin_name:<resistance> ...}

Define the package R/L/C value in that pin, which overwrites the default value listed in the [Package] section. Example:

```
* PIN_C_PIN={P3:2.9pF P2:3.4pF P1:2.0pF}
* PIN_L_PIN={P3:3.0nH P2:5.0nH P1:NA}
* PIN_R_PIN={P3:200m  P2:209m  P1:198m}
```

PIN_INV_PIN={pin_name:net_name ...}

Specify the corresponding inverting pin (and net) name for I/O output. Example:

```
* PIN_INV_PIN={P2:PADN}
```

PIN_MODEL_NAME={pin_name:model_name ...}

Specify the pin name to model name mapping. Note: POWER, GND and NC are reserved model names that will be set to model_type. Example:

```
* PIN_MODEL_NAME={VD33:POWER VSSPST:GND}
```

PIN_NET_NAME={pin_name:net_name ...}

Specify the IBIS pin name to netlist net (signal) name mapping. Example:

```
* PIN_NET_NAME={P3:DOUT P2:PAD P1:DIN}
```

PIN_TDELAY={pin_name:<time> ...}

This contains launch delays (in seconds) of the non-inverting pins relative to the inverting pins. Example:

```
* PIN_TDELAY={P2:1.0ns,NA,NA}
```

PIN_VDIFF={pin_name:<voltage> ...}

Specify the differential receiver threshold voltage between the inverting and non-inverting pins for Input or I/O model types. Example:

```
* PIN_VDIFF={P2:200m}
```

PULLDN_RES={pin_name:typ,min,max …}
(or PULLDOWN_RES)

Specify the pulldown R_load in Ohms for each pin.

PULLDN_VOLT={pin_name:typ,min,max …}
(or PULLDN_VOLTAGE or PULLDOWN_VOLT or PULLDOWN_VOLTAGE)

Specify the pulldown termination corner voltages for each pin.

PULLUP_RES={pin_name:typ,min,max …}

Specify the pullup R_load in Ohms for each pin. This should be consistent with the R_fixture.

PULLUP_VOLT={pin_name:typ,min,max …}
(or PULLUP_VOLTAGE)

Specify the pullup termination corner voltages for each pin.

RUN_DIR={/run_dir_typ[,/run_dir_min[,/run_dir_max]]}

Specifies between 1 to 3 directories where Liberate can find and execute Tcl commands associated with the corners: min, typ, and max. This command must be set between IBIS_BEGIN & IBIS_END statements.

Important: You must set the RUN_DIR when using a job scheduler such as LSF or Sun Grid Engine (SGE). If this is not set correctly, the path to the Tcl scripts sourced within _ibis_<cell>_<corner>.tcl will not be found.

For usage, reference the following examples below:

If RUN_DIR is set to {/run_dir_typ,/run_dir_min,/run_dir_max} then the generated scripts for the 3 corners will have their run directories set as follows:

```
_ibis_<cell>_typ.tcl  use:  /run_dir_typ
_ibis_<cell>_min.tcl  use:  /run_dir_min
_ibis_<cell>_max.tcl  use:  /run_dir_max
```

If only 2 RUN_DIR's are set {/run_dir_typ,/run_dir_min} but three corners are specified the run directories will be set as follows:

```
_ibis_<cell>_typ.tcl  use:  /run_dir_typ
```

```
_ibis_<cell>_min.tcl  use:  /run_dir_min    # These 2
_ibis_<cell>_max.tcl  use:  /run_dir_min    # repeated
```

If only 1 RUN_DIR is set {/run_dir_typ} but three corners are specified the run directory will be set as follows:

```
_ibis_<cell>_typ.tcl  use:  /run_dir_typ    #
_ibis_<cell>_min.tcl  use:  /run_dir_typ    # These 3 repeated
_ibis_<cell>_max.tcl  use:  /run_dir_typ    #
```

If RUN_DIR is not specified, then a **`set run_dir [exec pwd]`** is added in the run script ibis_<cell>_<corner>.tcl

SET_CONTROL_PARAM={<variables>}

SET_CONTROL_PARAM recognizes two variables: **no_header** and **append_file**. They accept the values 1 (or true) and 0 (or false). **append_file** has no effect if **no_header** is not set to 1 (or true). This is used to create an IBIS file with multiple [Model] sections for a single device. (See Creating an IBIS file with multiple [Model] sections for a single device.)

Example:

```
* SET_CONTROL_PARAM={no_header:1 append_file:1}
```

This will create the following line in igen_<CELL>.tcl :

```
write_ibis_file -no_header -append_file $file $ldbs $cell
```

SET_NET_GND={net_name:typ[,min[,max]]}

This specifies the net name of a pin and the ground voltage levels applied to each corner.

SET_NET_VDD={net_name:typ[,min[,max]]}

This specifies the net name of a pin and the supply voltage levels applied to each corner.

SET_POWER_NET_GND={supply_net_name:typ[,min[,max]]}

Specifies the name of a GND power source net name and the voltages applied to each corner.

SET_POWER_NET_VDD={supply_net_name:typ[,min[,max]]}

Specifies the name of a VDD power source net name and the voltages applied to each corner.

SET_TCL_VARIABLE={Tcl_variable_name:typ_val[,min_val[,max_val]]}

This command is to set harness (subckt cell name) variables which is to be used in **template_arc.tcl**.

Example:

```
* SET_TCL_VARIABLE={TERMRES:termres_typ,termres_min,termres_max}
* HARNESS=$TERMRES
```

Then in the generated (typical corner) char script file there will be the following:

```
set TERMRES termres_typ
```

and in the generated template_arc.tcl, there will be

```
define_cell \
    -input {A LOPWRA LOPWRB } \
    -output {PAD} \
    -pinlist {A LOPWRA LOPWRB PAD} \
    -harness $TERMRES \
...
```

The user should have a harness spice netlist file available.

Example: In the termres.sp file :

```
.subckt termres_typ pad_altos_out padn_altos_out
 R1 pad_altos_out  mid 50
 R2 padn_altos_out mid 50
 Vmid mid 0 1.21
.ends
```

In the predef.tcl file, there should be the following line: (note, there are 3 harness cell termres_typ/termres_min/termres_max for the three corners)

```
if {$corner == "typ"} {
    set TERMRES "termres_typ"
} elseif {$corner == "min"} {
    set TERMRES "termres_min"
} elseif {$corner == "max"} {
    set TERMRES "termres_max"
}
lappend spicefiles "termres.sp"
```

By doing so, the -harness argument in define_cell command in template_arc.tcl file can be functional.

```
define_cell \
        -input {A LOPWRA LOPWRB LPSEL PWRDN} \
        -output {PAD PADN} \
        -pinlist {A LOPWRA LOPWRB LPSEL PWRDN PAD PADN} \
        -harness $TERMRES \
        -delay delay_template_1x1 \
        -power power_template_1x1 \
        MyCell
```

Note, harness is not needed for liberate to characterize IBIS differential IO cells. This example is only for demo purpose.

NOTE: The rising dV is measure with an R_load (default 50 Ohm) termination to ground and the falling is measured with an R_load (default 50 Ohm) termination to Vdd. The following are voltages for pullup/down measurements and should be consistent with V_fixture.

Template generated :

The following Liberate commands are written to the generated template_arc.tcl file:

```
define_template
define_cell
define_pin_load
define_leakage
define_arc.
```

The following Liberate commands are written to the generated template_IBIS.tcl file:

```
ibis_define_header
ibis_define_waveform_template
ibis_define_component
ibis_define_model
ibis_define_pin,
```

The following commands take IBIS model name as leading variable:

```
MODEL_C_REF
MODEL_R_REF
MODEL_V_REF
MODEL_TYPE
MODEL_RELATED_PIN_NET
MODEL_ENABLE_PIN_NET
MODEL_WHEN
MODEL_VINL
MODEL_VINH
MODEL_VMEAS
```

The following commands take IBIS pin name as leading variable:

```
PULLUP_VOLT
PULLDN_VOLT
PULLUP_RES
PULLDN_RES
PIN_MODEL_NAME
PIN_NET_NAME
PIN_INV_PIN
PIN_INV_PIN_NET
PIN_R_PIN
PIN_L_PIN
PIN_C_PIN
PIN_L_FIXTURE
PIN_C_FIXTURE
PIN_VDIFF
PIN_TDELAY
```

The following commands take SPICE net name as leading variable:

```
SET_POWER_NET_VDD
SET_POWER_NET_GND
SET_NET_VDD
SET_NET_GND
```

Creating an IBIS file with multiple [Model] sections for a single device

The user may create an IBIS file from different sets of IBIS ldb files by using the options **-no_header**, and **-append_file**. The example below shows how these options are used in a series of three Tcl script to add [Model] data to a single IBIS file.

1. Prepare IBIS ldb files and 3 Tcl scripts

script_01.tcl

```
set cell "IOBUF12S"
set file "iobuf12s.ibs"
set ldbs { ibis_IOBUF12S_typ.01.ldb.gz ibis_IOBUF12S_min.01.ldb.gz \
    ibis_IOBUF12S_max.01.ldb.gz }
write_ibis_file $file $ldbs $cell
```

script_02.tcl

```
set cell "IOBUF12S"
set file "iobuf12s.ibs"
set ldbs { ibis_IOBUF12S_typ.02.ldb.gz ibis_IOBUF12S_min.02.ldb.gz \
    ibis_IOBUF12S_max.02.ldb.gz }
write_ibis_file -no_header -append_file $file $ldbs $cell
```

script_03.tcl

```
set cell "IOBUF12S"
set file "iobuf12s.ibs"
set ldbs { ibis_IOBUF12S_typ.03.ldb.gz ibis_IOBUF12S_min.03.ldb.gz \
    ibis_IOBUF12S_max.03.ldb.gz }
write_ibis_file -no_header -append_file $file $ldbs $cell
write_ibis_end $file
```

2. Run scripts sequentially

```
liberate script_01.tcl
liberate script_02.tcl
liberate script_03.tcl
```

3. Edit IBIS file to add [Model Selector] section manually and validate file integrity.

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 ldb as descibed 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 variables 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 variable so options appear in the expected location:

	Spice compatible	Native Spectre options	Transient control	Command line
Liberate variables	extsim_option extsim_leakage_option	extsim_deck_header	extsim_tran_append	extsim_cmd_option
Spectre options	gmin method rabsshort save	reltol	errpreset lteratio	-64 +altos +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 versoins.

Spectre Kernel Interface (SKI)

To enable the high-performance private API to Spectre (SKI), add these to baseline settings. The recommended version of MMSIM is 13.1 ISR5 or newer, in order to gain the full benefit of SKI.

Note: Liberate SKI Aging flow was only supported by Liberate13.1 ISR4 along with MMSIM13.1 ISR9 or later versions.

```
set_var ski_enable               1
```

Do not save the decks with SKI as the disk storage requirement is too high

```
set_var extsim_save_passed       none
```

Put SKI tmp files in an NFS partition instead of machine-local

```
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 (ie: 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 & 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
```

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
```

HSPICE

Accuracy settings for 28nm and below

```
set_var extsim_cmd  < path to HSPICE >
set_var extsim_leakage_option "accurate=1 brief=1 runlvl=6 method=gear
gmindc=1e-15 gmin=1e-15 kcltest=1"
set_var extsim_option "accurate=1 brief=1 runlvl=6 autostop gmindc=1e-15
gmin=1e-15"
```

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 variables 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 ecsm_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
set_var predriver_waveform_mode 1
set_var predriver_waveform_npts 17
set_var predriver_waveform_ratio 0.5

## CCS Timing Settings
set_var ccs_abs_tol 9e-13
set_var ccs_max_pts 50
set_var ccs_rel_tol 0.009
set_var ccs_voltage_tail_tol 0.955
set_var ccs_voltage_tail_trim_tol 0.999

## CCS Noise Settings
set_var ccsn_floating_init_mode 3

## CCS Power Settings
# Use 0 for correct CCSP behavior of bundle cells
set_var ccsp_related_pin_mode 1

## Compact CCS/CCSP
set_var ccs_base_curve_points 15
set_var ccs_base_curve_share_mode 2
set_var ccsp_base_curve_points  15
set_var ccsp_leakage_current_compensation_mode  1
```

D

Deprecated and Legacy Variables

This section lists all the variables that are either deprecated, or included for backward compatibility. We would like to discourage users from relying on these variables, and instead find alternate variables and settings to achieve the best results from Liberate.

Deprecated Variables

These variables are being phased out, and have been replaced by either new commands, new variables, or new behaviors of the tool.

bundle_count

<value> Number of bundles. Default: follow bundle_mem_limit.

This parameter is used to select a specific number of bundles. Bundle mode is enabled when bundle_mem_limit is exceeded or when this variable 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`, then do not set `bundle_count>0`.

This variable must be used before **char_library**.

default_capacitance

<min | avg | max> Use the minimum (min), average (avg) or maximum (max) rise/fall capacitance value as the default pin rise/fall capacitance. Default: max

Use this variable to set the desired selection criteria for the *capacitance* attribute.

Note: This variable has been deprecated. We recommend the set_default_group command be used instead of this variable.

This variable must be set before **char_library**.

default_group_method

< 0 | 1 >	Method to use for creating default groups. Default: 1 (use the whole table with the worst bit)

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**, then 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 variable does not apply to constraints. Constraint default groups are always selected on a bitwise basis. (Default)

Note: This variable is deprecated. We recommend using the set_default_group command be used instead of this variable.

This variable 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 variable has been deprecated. It is recommended that the set_default_group command be used instead of this variable.

This variable 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 variable has been deprecated. It is recommended that the set_default_group command be used instead of this variable.

This variable 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 variable has been deprecated. It is recommended that the set_default_group command be used instead of this variable.

This variable must be used before **char_library**.

default_unateness

<merge | separate> Control if default positive_unate and negative_unate timing groups should be merged into a single non_unate default timing group. Default: merge

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 variable 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.

Note: This variable has been deprecated. It is recommended that the set_default_group command be used instead of this variable.

define_em

-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 **define_em** command defines the cells and resistors and ports that will be characterized for electromigration models. 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. The Chapter 7 section on Electromigration Modeling describes these modes in detail. This command supports wildcards.

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}
```

rc_sort_mode

< 0 | 1 >	Enable sorting of RC elements. Default: 1

This variable 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 reverts to release 3.0p2 prior behavior where RC elements are never sorted. (Default: 1)

Note: This variable is deprecated with release 3.2p4 and will be ignored; RC's will always be sorted for consistency. Using this variable will generate a warning.

set_client

-n <number_of_clients>	
	Number of clients to use.
-dir <directory_name>{%N%U%P%S}	
	Directory for temporary files. (REQUIRED)
<Queue_name>	Name of the batch queue.

Important

The **-n** argument explained below 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."

The **-dir** argument 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.

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).

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** variable 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

-cells {list}	List of cell names
-resistor {resistor or output names}	List of resistor names or output pins
-type <"avg" | "rms">	Average, or RMS
<value>	IMAX value

Use this command to specify 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.

This command must be used before **write_library**. Examples:

```
# Set the man current for resistor R1 to 5mA
```

```
set_em_imax -type avg \
    -resistor { R1 } \
    -cells { inv and2 }
    5e-3
```

ski_alter_mode

< 1 | 2 > Controls method for generating SPICE decks used in constraint acquisition. Default: 2 (Alspice-compatible)

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 variable only has an effect if **ski_enable=1**. See External Simulator Options and Settings for recommended settings.

Note: This setting might affect performance. Generally ski_alter_mode=2 should be faster.

1. Use external SPICE-compatible settings (recommended for matching Spectre, HSPICE, Eldo, etc.)
2. Use Alspice-compatible settings (recommended for matching Alspice)

This variable must be used before **char_library**.

Backward Compatibility Variables

These variables invoke older behaviors of the tool. We generally discourage using these variables because many of the older algorithms have since been corrected or improved.

ccsn_active_ccr_recognition_mode

< 0 | 1> Specifies whether to do a more aggressive search for ccsn arcs. Default: 1

If this variable 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 variable must be used before **char_library**.

ccsn_check_valid_noise_prop

< 1 | 2> Checks if a CCB can propagate noise prior to characterization. Default: 1

Use this variable to enable an algorithm that screens the CCB for vectors that cause the CCB output to become tristated.

1: Check for a tristated CCB output.
2: Use this setting for backward compatibility to release 13.1ISR4 and prior releases.

This variable must be set before the **char_library** command.

ccsn_compatibility_mode

< 0 | 1 > Controls CCB topology checks for CCSN characterization. Default: 1.

Controls checks to more thoroughly analyze CCB topologies and electrical paths to identify and characterize CCSN stages at the timing and pin level.

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 variable must be used before **char_library**.

ccsn_compatibility_multi_corners

< 0 | 1 > Effects CCSN structure in library. Default: 1

0: Backward compatible to 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 > Controls algorithm for estimating the end time for CCSN simulations. Default: 1

Occasionally, the end time for voltage rise/fall CCSN simulations is not long enough and the final expected voltage was not reached. This variable causes the estimated end time to be extended for such simulations.

0: Backward compatible to 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)

This variable must be used before **char_library**.

ccsn_dc_sweep_mode

< 0 | 1 > For CCSN characterization compatibility when running Finesim as the external simulator. Default: 1

0: Finesim compatibility setting for CCSN characterization.
1: (Default)

ccsn_fanout_select_mode

< 0 | 1 > Sets CCSN groups on test pins. Default: 1

When this variable is set to **1**, Liberate includes CCSN groups on all test pins. Previously not all test pins had CCSN data.

This variable must be used before **char_library**.

ccsn_io_skip_channel_inputs

< 0 | 1 > Default: 1

This variable is used to change 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 cannot automatically drive ccsn characterization and the fallback approach of automatic vector determination is used.

0: No distinction about channel connectivity is made. This setting is used for backward compatibility to 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 variable must be used before **char_library**.

ccsn_input_xfr_probe_mode

< 0 | 1 >	Control CCSN modeling of input pins connected directly to pass-gates. Default: 1 (Recommended is also 1)

0: Do not generate ccsn_first_stage groups for input pins that connect directly to transistor pass gates. (Default)

1: Generate ccsn_first_stage_groups for input pins that connect directly to transistor pass gates. (Recommended)

This variable must be used before **char_library**.

ccsn_model_unbuffered_output

< 0 | 1 | 2 >	Specifies how to handle unbuffered outputs. Default: 2

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.

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. (Default and recommended).

This variable must be used before **char_library**.

ccsn_pin_unconditional

< 0 | 1 >	Set to 1 to stop Liberate from generating conditional pin-based CCSN stages. Default: 0

When set to 1, Liberate will only create a single CCSN pin stage when there are multiple CCSN pin stages possible. This can help 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.

The setting 1 should only be used for backwards-compatible purposes.

This variable must be used before **char_library**.

ccsn_prefer_two_sided_stages

< 0 | 1 | 2 > Set to prefer two-sided stages. Default: 2

Liberate will output two sided (stage_type: both) ccsn stages whenever possible. This helps to address errors in 2008 versions of LC (LC200809-SP3). Reset this variable to **0** to revert back to pre-v2.3 behavior. When set to **2**, Liberate will always try to generate **CCSN** stages that have *stage_type* set to *both*, regardless of whether the sensitization is feasible in the cell. Setting this variable to **2** should address any LBDB-898 Synopsys LC error messages.

This variable must be used before **char_library**.

ccsn_probe_mode

< 0 | 1 | 2> Method for selecting a probe node for noise measurements. Default: 2 (Balanced)

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.

0: Strict interpretation of the guidelines (Behavior of Liberate version 12.1.4 and earlier.)
1: Sorting method.
2: Balanced method. (Default and recommended)

This variable must be set before **char_library**.

ccsn_prop_retry_duration_incr

< 0 | 1 > Automatically extends the CCSN simulation duration on failures. Default: 1

Set this variable to 1 to automatically extend the simulation duration for CCSN measurements if the original simulation fails. If you see warnings regarding "Unable to obtain reasonable CCS noise propagated waveform", then set this variable and re-run characterization.

The default and recommended setting is 1.

This variable must be set before **char_library**.

ccsn_redundant_pin_stages

< 0 | 1 > Set to duplicate certain CCSN stages on both arc and pin. Default: 0 (Do not duplicate)

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.

0: Do not allow redundant stages (Default and recommended)
1: Allow redundant stages (3.2p4_lcs behavior)

This variable must be used before **char_library**.

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

Intended for backward compatibility only.

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**

This variable must be used before **write_library**.

char_mos_term_cap_ski_mode

< 0 | 1> Default: 1

Automates the MOS terminal capacitance calculation when using the SKI interface.

0: Follow the setting of the `char_mos_term_cap` variable. This setting is used for backward compatibility to 13.1 ISR4 and prior releases.
1: When the SKI interface is enabled (see `ski_enable`), then override the setting of char_mos_term_cap with a value of 1.

This variable 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

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.

0: For backward compatibility to pre-3.2 versions. May result in more search iterations. (Not recommended.)
1: For backward compatibility to pre-13.1.3 version. May result in more search iterations. (Not recommended.)
2: The search tolerance follows constraint_search_time_abstol and constraint_search_time_reltol. (Default)

This variable must be used before **char_library**.

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 variable must be used before **char_library**.

default_power_subtract_hidden_mode

< 0 | 1 | 2 >	Controls `subtract_hidden_power` algorithm. Default:2

This variable is used to control the algorithm used when `subtract_hidden_power` is enabled.

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 variable must be used before **write_library**.

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`)

This variable is used to control `timing_type` attribute for default `three_state_enable` timing groups.

0: The default arc timing_type for three_state_enable arcs will be set to 'combinational'. This setting is provided for backward compatibility to release LIBERATE 13.1 ISR4 and prior releases.
1: the arc timing attribute uses `three_state_enable` (Default).

This variable must be set prior to the **char_library** command.

def_arc_delay_metric_mode

< 0 | 1 >	For backward compatibility with `define_arc -metric`. Default=1

The `define_arc -metric` option should always control the constraint measurement metric.

0: Revert to release **12.1** and prior behavior where under certain conditions, the Liberate internal algorithms would override the user-specified `define_arc -metric`.
1: (Default and recommended.)

This variable must be used before **char_library**.

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 to 0 to revert to release 2.5 and prior behavior, where Liberate reports the time point when output current drops below the current degradation threshold.

This variable must be used before **char_library**.

driver_waveform_lib_mode

< 0 | 1 >	Affects format of normalized_driver_waveform. Default: 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 variable for backward compatibility.

0: Format for normalized_driver_waveform used with version **3.2p3** or earlier.
1: Standard format for normalized_driver_waveform. (Default and recommended.)

This variable must be used before **write_library**.

ecsmp_invert_gnd_current

< 0 | 1 > Inverts current values for ecsm_current_waveform groups. Default: 1 (Invert current.)

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's power tools prefer to have ground currents reported as positive values. This variable controls whether or not to invert the current on grounds, which are reported in index_1 of ecsm_current_waveform groups.

0: Do not invert currents.
1: Inverts the current values for compatibility with EPS. (Default)

This variable must be used before **char_library**.

extsim_model_include_multi_vector_mode

< 0 | 1 > For backward compatibility. Uses 3.1 (and prior) algorithm Default: 1

Version 3.1p1 corrects an issue handling constraint-related vectors in the EMI flow when `extsim_flatten_netlist` is set to 0.

0: Backward compatibile to release **3.1**.
1: (Default)

extsim_save_driver

<0 | 1> Save SPICE decks for failing active driver simulations. Default: `1`

This variable is used to enable the saving of simulation decks used by the **set_driver_cell**1 command to characterize the active driver output waveform simulation decks.

0: Specifies not to save any active driver simulation decks. This option 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` variables. If the saving of decks is not enabled using these variables, no decks are saved.

This variable must be set before the **char_library** command.

floating_node_consistency_check

< 0 | 1> This variable enables floating node consistency checks introduced in 3.1p2. Default: 1

This variable should only be disabled to reproduce the inconsistent floating node handling behavior of Liberate versions 3.1p2 and earlier for regression purposes. Note that this behavior has known issues and is not recommended for general use.

0: Backward compatible to release **3.1p2**. Disables floating node consistency checks.
1: Enable the floating node consistency checks. (Default and recommended.)

This variable must be used before **char_library**.

ignore_dummy_fets

< 0 | 1 | 2 | 3> Default: 2

Use this variable to determine how Inside View handles non-functional or "dummy" transistors such as PODE devices. 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 exponential increase in the number of vectors to evaluate/simulate for no real purpose.

Setting this variable to `0` causes Inside View to consider the dummy devices to have a functional impact. All other settings ignore any functional impact during Inside View's preprocessing and vector generation, and differ only in how initial conditions (.IC) are set for dummy devices with floating gates.

Use this variable to determine how Inside View handles non-functional or "dummy" parasitic transistors such as PODE devices. 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, causing an increase in the runtime for no real purpose.

Setting this variable to `0` causes Inside View to consider the dummy devices to have a functional impact and is provided for backward compatibility. All other settings ignore the functional impact of these parasitic devices during Inside View's preprocessing and vector generation, and differ only in how initial conditions (.ic) are set for the dummy devices with floating gates.

0: Consider functional impact of dummy devices. Should only be used for backward compatibility for 13.1 and previous 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 will be 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.

This variable must be used before **char_library**.

non_seq_probe_mode

< 0 | 1 >	Controls the probe selection mode for *nonseq_setup/hold.* Default: 1

Specifies the non_seq probing order. When set to 1, Liberate will try to honor **constraint_output_pin.**

0: Compatibility with releases prior to version 2.2.
1: Compatibility with release 2.2 and later. (Default)

This variable must be used before **char_library**.

power_multi_vector_mode

< 0 | 1 >	Corrects issue of choosing wrong vector. Default: 1

Related to an issue where Liberate could choose a wrong vector when an arc contained multiple vectors and the `voltage_map > 0`. This could produce power results that may not be worst case.

0: Backward compatible to release 3.1p1.
1: Corrects issue. (Default)

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)

Backward compatibility option. 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.

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 variable must be used before **char_library**.

reset_negative_power_mode

< 0 | 1 > Controls the behavior for fixing negative power values. Default: `1` (Follow `reset_negative_power`).

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.

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)

This variable must be set before **char_library**.

set_pin_slew_threshold_mode

< 0 | 1 > Controls handling of slew threshold derating for power. Default: `1`

This variable enables correct processing of the **set_pin_threshold** command settings.

0: Scale the `index_*` and value in the pin group specified. (Only for backward compatibility to release 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)

This variable must be set before **char_library**.

switch_cell_infer_unateness_from_ldb

< 0 | 1 > Controls determination of unateness for switch cells. Default: 1

For backward compatibility when writing out a library from an older ldb (prior to v3.2). Only relevant to switch cells.

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. (Default and recommended.)

This variable must be used before **write_library**.

tristate_pin_cap_use_arc

< 0 | 1 | 2 > Default: 2

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.

0: For backwards compatibility.
1: New method only.
2: New method with fall back to original method if required. (Default)

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 variable must be set before **write_library**.

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** variable is set. Default: `1`

0: When the **write_logic_function** variable is set to `0`, the arcs with a *timing_type* of *preset* and *clear* will be changed to a *timing_type* of *combinational*. Use this setting for backward compatibility to 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** variable.

This variable must be set before **write_library**.

voltage_map_ldb_char_mode

< 0 | 1 > Corrects erroneous supply name in library. Default: 1

Applicable to circuits that do *not* use VDD, while running `read_library` / `write_library` flow.

0: Backward compatible to version **3.2p3** and earlier.
1: Corrects issue where default supply name was erroneously written to library when `voltage_map=2`. (Default)

This variable must be used before **write_library**.

Legacy Debug Variables

These variables may or may not work. This is just a list of variables 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 variable 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.

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
```

This variable must be used before **char_library**.

ccs_retry_mode

< 0 | 1 | 2 | 3 > Enables CCS simulation retry methodologies. Default: 0 (Disabled)

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 variable controls the methodology and criteria used for detecting failures.

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.

Note that 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 may 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.

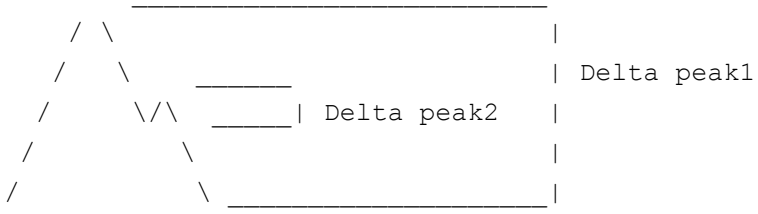
This variable must be used before **char_library**.

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 variable, 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 variable must be used before **char_library**.

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 variable must be used before **char_library**.

Example: set_var ccs_retry_voltage_tail_tol 0.951

extsim_ccs_retry_option

<"options"> Options to be used for CCS timing re-characterization 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 variable 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 variable must be used before **char_library**.

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 variable must be used before **char_library**.

Example:

Set conservative mode for Spectre for CCS retry simulations

set_var extsim_ccs_retry_tran_append "errpreset=conservative"

COMMAND

set_ccs_retry_thresholds

-rise {list} List of voltage thresholds specified as a percentage in decimal. (i.e: .5 = 50%)

-fall {list} List of voltage thresholds specified as a percentage in decimal. (i.e: .5 = 50%)

This command allows users to override the default behavior of Liberate with respect to voltage thresholds checked during CCS-T characterization.

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, then Liberate will check 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, then the simulation will be retried with more accurate options.

This command must be used before **char_library**.

Example 1:

```
# 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]
```

E

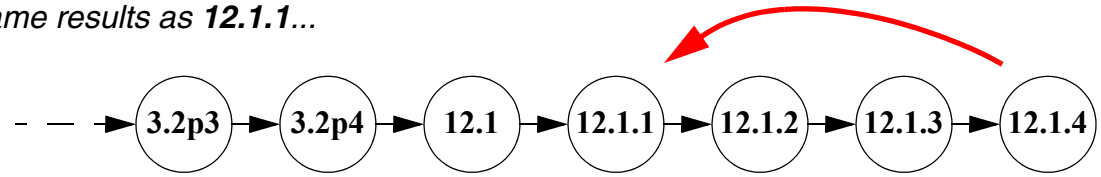
Variables to Use When Qualifying and Migrating Between Versions

These variables allow you to run your current version of Liberate and get the same results as an earlier version.

The variables 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's that go back to the version you need. (See example.)

Note: Default settings are generally changed to comply with updated specifications, requirements from downstream tools, or to correct significant issues. These backwards-compatible settings are 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's 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 init_constraint_period_binning_mode 0
...

# 12.1.1 to 12.1
set_var ccs_segmentation_effort 0x1
set_var ccsn_compatibility_multi_corners 0
...
```

```
# 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
```

F

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
IBIS	Input/output Buffer Information Specification
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 Kernal 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 (Scale?) Integrated Circuit
VITAL	VHDL Initiative Towards ASIC Libraries