

Library Quality Assurance System User Guide

Version O-2018.06, June 2018

SYNOPSYS®

Copyright Notice and Proprietary Information

©2018 Synopsys, Inc. All rights reserved. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

Destination Control Statement

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

Disclaimer

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

Trademarks

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

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

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

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

Copyright Notice for the Command-Line Editing Feature

© 1992, 1993 The Regents of the University of California. All rights reserved. This code is derived from software contributed to Berkeley by Christos Zoulas of Cornell University.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- 1.Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- 2.Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- 3.All advertising materials mentioning features or use of this software must display the following acknowledgement:

This product includes software developed by the University of California, Berkeley and its contributors.

- 4.Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright Notice for the Line-Editing Library

© 1992 Simmule Turner and Rich Salz. All rights reserved.

This software is not subject to any license of the American Telephone and Telegraph Company or of the Regents of the University of California.

Permission is granted to anyone to use this software for any purpose on any computer system, and to alter it and redistribute it freely, subject to the following restrictions:

- 1.The authors are not responsible for the consequences of use of this software, no matter how awful, even if they arise from flaws in it.
- 2.The origin of this software must not be misrepresented, either by explicit claim or by omission. Since few users ever read sources, credits must appear in the documentation.
- 3.Altered versions must be plainly marked as such, and must not be misrepresented as being the original software. Since few users ever read sources, credits must appear in the documentation.
- 4.This notice may not be removed or altered.

Copyright Notice for the Nonlinear Optimization Library

© 2011 by Massachusetts Institute of Technology.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

1. The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
2. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Contents

About This Guide	xiv
Customer Support	xvi
1. Reading and Compiling Libraries	
Reading In Libraries	1-2
Compiling a Library With CCS Signal Integrity Information	1-2
Reading Job Completion Record (JCR) Information	1-3
Library Screener Checks	1-4
Checks for 10 nm Libraries	1-5
Operating Condition Checks	1-7
On-Chip Variation Checks	1-7
LVF Checks For Cell Delay, Transition, and Constraint	1-8
LVF-Moments OCV Checks	1-10
Retain Arc OCV Checks	1-11
Characterization Configuration Checks	1-12
Driver Waveform Checks	1-13
Library-Level Checks	1-13
Cell-Level Checks	1-14
Timing Arc Sensitization Checks	1-14
Sensitization Group Checks	1-15
Cell-Level Checks	1-15
Timing-Level Checks	1-15
Phase-Locked Loop Checks	1-16
Physical Variant Cell Checks	1-17
Decoupling Capacitor, Filler, and Tap Cell Checks	1-17

Programmable Driver Type Checks	1-17
Statetable Checks	1-18
Scan Cell Checks	1-19
test_cell Group Checks	1-19
Scan Cell Timing Arc Checks	1-19
Multibit Scan Cell Checks	1-20
Checks for Multibit Scan Cells With Internal Scan Chains	1-20
Scan-Enabled LSSD Cell Checks	1-21
Generic Integrated Clock-Gating Cell Checks	1-21
Advanced Low-Power Screener Checks	1-22
PG Pin Library Checks	1-22
Partial PG Pin Cell Checks	1-24
Feedthrough Cell Checks	1-24
Overdrive and Underdrive Voltage Checks	1-24
Unconnected Pin Checks	1-25
Level-Shifter Cell Checks	1-25
Isolation Cell Checks	1-27
Clamp Function Checks	1-29
Internal Isolation Checks for Macro and Pad Cells	1-29
Switch Cell Checks	1-30
Retention Cell Checks	1-32
Always-On Cell Checks	1-35
Antenna-Diode Cell Checks	1-36
Composite Current Source (CCS) Screener Checks	1-37
CCS Driver Model Checks	1-37
Two-Segment Receiver Capacitance Model Checks	1-38
Multisegment Receiver Capacitance Model Checks	1-40
Current Polarity and Full Voltage Swing Checks	1-40
Advanced CCS Screener Checks	1-43
Compact CCS Timing Model Checks	1-43
CCS Signal Integrity Modeling Checks	1-45
Referenced CCS Noise Modeling Checks	1-48
CCS Power Modeling Checks	1-51
Leakage Current Syntax Checks	1-51
Gate Leakage Modeling in Leakage Current Syntax Checks	1-53
Intrinsic Parasitic Model Syntax Checks	1-53
Parasitics Model Syntax Checks	1-55
Dynamic Power Model Syntax Checks	1-55
Power and Ground Current Syntax Checks	1-56
Dynamic Current Syntax Checks	1-56

Compact CCS Power Modeling Checks	1-60
Electromigration Checks	1-61
NLDM Timing Data and NLDM Noise Data Screener	1-61
Nonlinear Delay Noise Variables	1-63
NLDM Noise Data Screener	1-64
Basic Checks	1-64
Recommended Checks	1-64
Extrapolation Checks	1-65
Current Polarity	1-66
Negative Table Values	1-66
Advanced Checks	1-66
2. Generating Library Reports	
Overview	2-3
Generating Default Reports	2-3
Operating Condition Information	2-6
Generating Specific Reports	2-6
Timing Reports	2-7
Library Defaults	2-7
Library-Level Templates	2-8
Lookup Table Template	2-8
Template Examples	2-8
Cell-Level Data	2-9
Syntax	2-9
Pin Attributes	2-9
Operating Conditions	2-10
Pin-Level Information	2-10
Constraints	2-10
Constraint Syntax	2-11
Constraint Table Syntax	2-13
Delay Reports	2-14
Linear Combinational Delay	2-14
Piecewise Linear Combinational Delay	2-15
Nonlinear Combinational Delay	2-15
Rise and Fall Values	2-15
Linear Sequential Delay	2-16
Piecewise Linear Sequential Delay	2-17
Nonlinear Sequential Delay	2-17

Pin Capacitance Ranges	2-17
Cell Degradation Design Rule	2-17
Timing Arcs Report	2-18
Timing Label Reports	2-18
Timing Report Statistics	2-19
CCS Timing Model Information	2-19
High-Level Information	2-19
Detailed Information	2-19
Converting CCS Data to NLDM Data	2-20
Interdependent Setup and Hold Model Information	2-20
Sensitization Model Information	2-21
Predriver Model Information	2-22
Feedthrough Timing Arc Information	2-23
Mode Pin Information	2-23
On-Chip Variation Model Information	2-24
Noise Reports	2-25
CCS Noise Model Reports	2-28
High-Level Information	2-28
Detailed Information	2-28
Power Reports	2-28
Library Defaults	2-29
Library-Level Templates	2-29
Library-Level Multiple Power Supply Information	2-31
Cell-Level Information	2-31
Pin-Level Information	2-32
Generic Integrated Clock Gating (ICG) Cell Model Information	2-34
Power and Ground (PG) Pin Information	2-34
Partial PG Pin Cell Information	2-36
Feedthrough Cell Information	2-38
Unconnected Pin Information	2-39
Silicon-on-Insulator Cell Information	2-39
Switch Cell Information	2-39
Retention Cell Information	2-40
Always-On Cell Information	2-42
Antenna-Diode Cell Information	2-42
Level-Shifter Cells and Isolation Cells	2-43

Partially Powered-Down Isolation Cells	2-44
Macro Cell Isolation Information	2-44
CCS Power Model Information	2-45
Nonlinear Power Model Information	2-46
Analog Signal Attribute Information	2-47
Pad Cell Attribute Information	2-47
Electromigration Reports	2-47
Library Defaults	2-47
Library-Level Templates	2-47
Pin-Level Information	2-48
Functionality Reports	2-49
Statetable Report	2-49
Multibit Report	2-51
Programmable Driver Type Model Information	2-52
Physical Library Reports	2-52
FPGA Reports	2-55
Library Defaults	2-55
Parts Information	2-55
Cell-Level Information	2-56
User-Defined Data Reports	2-56
Job Completion Record (JCR) Reports	2-59
Comprehensive Reports	2-60
Cell Type Information	2-61
Generating Reports in HTML Format	2-61
Prioritizing and Customizing Messages	2-64
3. Library Checking	
Library Checking Overview	3-2
Checking Between Logic Libraries	3-5
General Logic Checks	3-5
library Group	3-5
cell Group	3-5
Pin and PG Pin Groups in the Cells	3-6
Timing Arcs	3-6

Special Logic Checks	3-6
Comparison Checks	3-10
Checking Specific Groups and Attributes	3-11
Analyzing Groups With Specific Attribute Settings	3-13
Comparing Libraries by Group Order	3-14
Maximum Transition and Maximum Capacitance	3-14
Scaling Checks	3-15
Checking Library Data Consistency	3-15
Checking the Scaling Library Group	3-20
Limitations.	3-21
Multivoltage and UPF	3-21
UPF Checks for Level-Shifter Cells	3-22
UPF Checks for Isolation Cells	3-23
UPF Checks for Switch Cells	3-24
UPF Checks for Retention Cells	3-25
UPF Checks for Always-on Cells	3-25
UPF Checks for PG Pin and Partial PG Pin Cells	3-25
UPF Checks for Substrate Bias Cells	3-26
Power Optimization Checks.	3-26
Multicorner-Multimode	3-27
Specifying Tolerances	3-28
Library Validation	3-29
Validating Timing.	3-30
Validating CCS Noise Models Against NLDM.	3-31
Compensation for the PrimeTime Tool.	3-32
Logical Equivalence Checks for when Conditions.	3-32
Inverter and Buffer Cells	3-33
Library Analysis	3-33
Value Range Analysis of Attributes	3-33
Trend Analysis for Single Characterization Tables	3-35
Variation-Aware Analysis	3-39
Table Bounds, Slope, and Index Analysis.	3-42
Sensitivity and Voltage Range Analysis	3-47
NLDM Index Spacing Analysis Using Interpolation.	3-51
Liberty Variation Format Analysis	3-54
CCS Power Analysis.	3-57
Checking CCS Consistency, Sensitivity, and Voltage Range Together	3-62
Checking for 10 nm Logic Libraries	3-62
Checking ETM Libraries	3-62

Qualifying ETM Libraries	3-63
Grouping ETM Libraries for Specific Checks	3-64
Reporting ETM Libraries	3-65
Qualifying Model File Libraries	3-66
Checking Individual Logic Libraries	3-66
Checking Physical Libraries	3-67
Checking Between Logic and Physical Libraries	3-67
Default Checks	3-68
Missing Cells Report	3-69
Missing and Mismatched Pins Report	3-69
Specific Checks	3-70
Generating Tcl Script to Fix Inconsistencies	3-72
check_library Reports	3-73
Report Formats	3-74
Default Text Report	3-74
Reporting Missing Groups	3-76
Reporting Line Numbers From Library Source Files	3-76
Specifying a Report in CSV Format	3-78
Specifying a Report in HTML Format	3-81
Specifying a Report in SQL Format	3-82
SQL Syntax for Data Analysis	3-83
Using the SQLite Browser	3-84
Analyzing Library Data Using SQL Queries in SQLite Browser	3-88
Analyzing Library Data Using SQL-Based Scripts	3-94

4. Library Quality Assurance Flows

Flows for Timing Models	4-2
Using the read_lib Command to Check Delay and Constraint Models	4-2
Using the check_library Command to Check Delay Models	4-2
Using the check_library Command to Check Constraint Models	4-3
Using the report_lib Command to Generate Timing Reports	4-4
Flows for CCS Noise Models	4-5
Using the read_lib Command to Check CCS Noise Data	4-5
Using the report_lib Command to Report CCS Noise Data	4-6
Flows for Power Models	4-6
Using the read_lib Command to Check NLPM and CCS Power Data	4-7

Using the report_lib Command to Report Power Data	4-7
Flows for UPF Models	4-8
Using the read_lib Command to Check UPF Data	4-8
Using the check_library Command to Check UPF Data	4-9
Using the report_lib Command to Report PG Pin Data	4-9
Flows for Scaling Libraries	4-9
Flows for Multicorner-Multimode Libraries	4-10

Preface

This preface includes the following sections:

- [About This Guide](#)
- [Customer Support](#)

About This Guide

The *Library Quality Assurance System User Guide* provides information about reading and compiling libraries using the `read_lib` command, the `read_lib` library screener checks, generating library reports using the `report_lib` command, and checking logic and physical libraries using the `check_library` command.

Audience

The *Library Quality Assurance System User Guide* is intended to assist library developers, designers, and electronics engineers. The *Library Quality Assurance System User Guide* is targeted for use with all Galaxy tools.

Related Publications

The *Library Quality Assurance System User Guide* supports the use of Library Compiler™, SiliconSmart®, and similar tools. For additional information about the Library Compiler tool, see the documentation on SolvNet at the following address:

<https://solvnet.synopsys.com/DocsOnWeb>

You might also want to see the other available Library Compiler documentation:

- *Library Compiler User Guide* describes the Library Compiler software, the `lc_shell` command syntax, how to build logic libraries and define cells, and how to model timing, signal integrity, and power in logic libraries, and how to prepare physical libraries.
- *Synopsys Logic Library Reference Manual* provides the syntax of the group statements that identify the characteristics of logic libraries.

Release Notes

Information about new features, changes, enhancements, known limitations, and resolved Synopsys Technical Action Requests (STARs) is available in the *Library Compiler Release Notes* in SolvNet.

To see the *Library Compiler Release Notes*,

1. Go to the Download Center on SolvNet located at the following address:
<https://solvnet.synopsys.com/DownloadCenter>
2. Select Library Compiler, and then select a release in the list that appears.

Conventions

The following conventions are used in Synopsys documentation.

Convention	Description
Courier	Indicates syntax, such as <code>write_file</code> .
<i>Courier italic</i>	Indicates a user-defined value in syntax, such as <code>write_file design_list</code> .
Courier bold	Indicates user input—text you type verbatim—in examples, such as <code>prompt> write_file top</code>
[]	Denotes optional arguments in syntax, such as <code>write_file [-format fmt]</code>
...	Indicates that arguments can be repeated as many times as needed, such as <code>pin1 pin2 ... pinN</code>
	Indicates a choice among alternatives, such as <code>low medium high</code>
Ctrl+C	Indicates a keyboard combination, such as holding down the Control key and pressing C.
\	Indicates a continuation of a command line.
/	Indicates levels of directory structure.
Edit > Copy	Indicates a path to a menu command, such as opening the Edit menu and choosing Copy.

Customer Support

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

Accessing SolvNet

SolvNet includes a knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. SolvNet also gives you access to a wide range of Synopsys online services including software downloads, documentation, and technical support.

To access SolvNet, go to the following address:

<https://solvnet.synopsys.com>

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

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

Contacting the Synopsys Technical Support Center

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

- Open a support case to your local support center online by signing in to SolvNet at <https://solvnet.synopsys.com>, clicking Support, and then clicking “Open A Support Case.”
- Send an e-mail message to your local support center.
 - E-mail support_center@synopsys.com from within North America.
 - Find other local support center e-mail addresses at <https://www.synopsys.com/support/global-support-centers.html>
- Telephone your local support center.
 - Call (800) 245-8005 from within North America.
 - Find other local support center telephone numbers at <https://www.synopsys.com/support/global-support-centers.html>

1

Reading and Compiling Libraries

You use the `read_lib` command to load a logic library source file (.lib) into the Library Compiler tool and compile it to the Synopsys database format (.db). The `read_lib` command automatically performs basic syntax checks before it writes out the compiled library. If a required argument or attribute is missing, or if the syntax is incorrect, the Library Compiler tool issues a warning message or an error message, as applicable.

This chapter includes the following sections:

- [Reading In Libraries](#)
- [Library Screener Checks](#)

Reading In Libraries

The `read_lib` command loads a logic library source file (.lib) into the Library Compiler tool and compiles it to the Synopsys database format (.db). When you load a .lib file, the Library Compiler tool checks for a Library-Compiler license. If the license is missing, the library is still loaded, but all the functional information is removed; that is, all cells are black boxes and optimization is disabled. If the license is present, the Library Compiler tool releases the license when the `read_lib` command finishes loading the .lib file.

The `read_lib` command automatically performs basic syntax checks before it writes out the compiled library. If a required argument or attribute is missing, or if the syntax is incorrect, the Library Compiler tool issues a warning message or an error message, as applicable. In addition, the `read_lib` command automatically performs screener checks for the following data: timing, power, noise, variation-aware, scaling, multicorn-multimode, and IEEE 1801, also known as Unified Power Format (UPF). For more information about the `read_lib` command screener checks, see [“Library Screener Checks” on page 1-4](#).

To read and compile a library, such as a cmos.lib logic library, run the `read_lib` command at the Library Compiler prompt, as shown:

```
lc_shell> read_lib cmos.lib
```

The arguments and options used with the `read_lib` command are as follows:

Arguments

filename

Name of the logic library; *filename* is a string.

`-test_model`

Specifies a set of one or more Core Test Language (CTL) files as a test model of cells in a library.

`-imported_jcr jcrfile`

Specifies the name of the job completion record (JCR) file to be imported. The job completion record file is an ASCII file.

`-model_type {lvf | ccs_power} model_type_name`

Specifies a list of model file names. The valid model types are `lvf` and `ccs_power`.

Compiling a Library With CCS Signal Integrity Information

When you compile a library containing CCS noise information with the `read_lib` command in the Library Compiler tool (after screening data without finding any errors), the Library Compiler tool extracts the required information from the CCS noise data in the library.

Often you only need to screen the CCS noise information in the libraries to see if the data is acceptable. This involves including or excluding the noise compilation. The `lc_disable_ccs_noise_extraction` environment variable enables you to turn on or off the compilation of CCS noise information.

By default, `lc_disable_ccs_noise_extraction` is set to 0, which means that the Library Compiler tool screens and extracts CCS noise information in the libraries with the `read_lib` command.

If `lc_disable_ccs_noise_extraction` is set to 1 before reading in a library with the `read_lib` command, the CCS noise information is screened but not extracted into the compiled library database. As a result, the compiled library database does not contain the CCS noise information.

Reading Job Completion Record (JCR) Information

A job completion record (JCR) is a summary of the tasks performed by a Synopsys product during a particular task in a design flow. The job completion record includes information, such as the error and warning messages generated, the number of occurrences of the error and warning messages, and the CPU and memory usage. Other tools can use the job completion record to check the quality of these tasks.

To enable the Library Compiler tool to read, store, and report the job completion record information, set the `lc_jcr_enable_read_lib` variable to `true`, as shown. The default is `false`.

```
prompt> set lc_jcr_enable_read_lib true
```

When you set this variable to `true`, the Library Compiler tool is enabled to

- Read in the job completion record file of a characterization tool, such as the Liberty NCX or the SiliconSmart tool, and store this information in the `.db` file.

To read in the characterization job completion record file, use the `-imported_jcr` option with the `read_lib` command. When you specify the `read_lib -imported_jcr jcrfile` command, the Library Compiler tool checks if the characterization job completion record file exists and is accessible. If the characterization job completion record file does not exist or is inaccessible, the tool generates an error message before reading in the `.lib` file.

- Generate its own job completion record file and store it in the `.db` file.

To exclude proprietary information from the Library Compiler job completion record, set the `lc_jcr_exclude_proprietary_info` variable to `true`, as shown. The default is `false`.

```
prompt> set lc_jcr_exclude_proprietary_info true
```

When you set this variable to `true`, the Library Compiler tool does not include the .lib file path and host name in its own job completion record.

- Report both the characterization and its own job completion record information.

For more information about generating job completion record reports, see [Chapter 2, “Generating Library Reports.”](#)

Library Screener Checks

The `read_lib` command automatically performs screener checks for timing, power, noise, variation-aware, IEEE 1801, scaling, multicorner-multimode data and reports a warning message or error message as applicable.

The following sections provide information about the available checks:

- [Checks for 10 nm Libraries](#)
- [Operating Condition Checks](#)
- [On-Chip Variation Checks](#)
- [Characterization Configuration Checks](#)
- [Driver Waveform Checks](#)
- [Timing Arc Sensitization Checks](#)
- [Phase-Locked Loop Checks](#)
- [Physical Variant Cell Checks](#)
- [Decoupling Capacitor, Filler, and Tap Cell Checks](#)
- [Programmable Driver Type Checks](#)
- [Statetable Checks](#)
- [Scan Cell Checks](#)
- [Feedthrough Cell Checks](#)
- [Generic Integrated Clock-Gating Cell Checks](#)
- [Advanced Low-Power Screener Checks](#)
- [Composite Current Source \(CCS\) Screener Checks](#)
- [Advanced CCS Screener Checks](#)
- [CCS Signal Integrity Modeling Checks](#)

- [CCS Power Modeling Checks](#)
- [Electromigration Checks](#)
- [NLDM Timing Data and NLDM Noise Data Screener](#)
- [NLDM Noise Data Screener](#)

Checks for 10 nm Libraries

The `read_lib` command performs some specific checks on 10 nm logic libraries. To enable the checks, specify:

```
lc_shell> set lc_enable_10nm_mode true
```

The `read_lib` command checks the numerical values of capacitances including the pin-level `capacitance`, `rise_capacitance`, and `fall_capacitance` attributes and generates the LBDB-954 error message if any of the values is zero. An absolute capacitance of less than 10^{-6} units is considered zero.

The exceptions are the following attributes and groups, for which the tool generates the LBDB-954w warning message if any of the capacitance values are zero:

- The `receiver_capacitance1_rise`, `receiver_capacitance1_fall`, `receiver_capacitance2_rise`, and `receiver_capacitance2_fall` groups in the `receiver_capacitance` group at both the pin and timing levels.
- The `millier_cap_rise` and `millier_cap_fall` attributes in the `ccsn_first_stage` and `ccsn_last_stage` groups at both the pin and timing levels.
- A library or cell where the `is_macro_cell`, `is_memory_cell`, `interface_timing`, `pad_cell`, `auxiliary_pad_cell` attributes are set to true, or the `power_cell_type` is macro, or the `timing_model_type` attribute is specified.
- A library or cell with unconnected pins.

The `read_lib` command checks the `receiver_capacitance` group capacitances specified using the following attributes and groups, and generates the LBDB-955 warning message if any of the following rules are not met:

- The values of the pin and timing level `receiver_capacitance1_rise` and `receiver_capacitance1_fall` groups must be less than twice the pin-level `rise_capacitance` and `fall_capacitance` attribute values respectively.
- If the `rise_capacitance` and `fall_capacitance` attribute values are not defined at the pin level, the values of each of the `receiver_capacitance1_rise` and `receiver_capacitance1_fall` groups must be less than twice the pin-level capacitance attribute value.

- The values of the pin and timing level `receiver_capacitance2_rise` and `receiver_capacitance2_fall` groups must be less than five times the pin-level `rise_capacitance` and `fall_capacitance` attribute values respectively.
- If the `rise_capacitance` and `fall_capacitance` attributes are not defined at the pin level, the values of each of the `receiver_capacitance2_rise` and `receiver_capacitance2_fall` groups must be less than five times the pin-level `capacitance` attribute value.

Note:

If the `receiver_capacitance1_rise`, `receiver_capacitance1_fall`, `receiver_capacitance2_rise`, and `receiver_capacitance2_fall` groups are defined within the `timing` group of an input pin, their capacitance values are compared against the `capacitance` attribute value of the input pin.

If the `receiver_capacitance1_rise`, `receiver_capacitance1_fall`, `receiver_capacitance2_rise`, and `receiver_capacitance2_fall` groups are defined within the `timing` group of an output pin, their capacitance values are compared against the `capacitance` attribute value of the related pin. The related pin is defined in the `timing` group using the `related_pin` attribute.

The `read_lib` command checks the Miller capacitances specified using the following attributes and groups, and generates warning messages if any of the following rules are not met:

- Both the `millier_cap_rise` and `millier_cap_fall` attribute values defined in the `ccsn_first_stage` group must be less than the pin-level `capacitance` attribute value and greater than 10 percent of the `capacitance` attribute value. Otherwise, the LBDB-955 warning message is generated.
- The ratio of the `millier_cap_rise` to the `millier_cap_fall` attribute value must be less than five. Otherwise, the LBDB-956 warning message is generated.

The `read_lib` command checks driver waveform data and generates the LBDB-957 warning message if any of the following rules is not met:

- The library must have the `normalized_driver_waveform` group.
- For a library-level `normalized_driver_waveform` group, the cell or pin level `driver_waveform`, `driver_waveform_rise`, or `driver_waveform_fall` attribute must be defined.

Note:

The `read_lib` command checks CCS noise models of pass gates, and does not generate the LBDB-613 error message if you specify only the `dc_current` group in a `ccsn_first_stage` or `ccsn_last_stage` group.

Operating Condition Checks

The Library Compiler tool checks the `operating_conditions` group syntax and issues an error message if any of the following conditions occur:

- The `operating_conditions` group is not defined in a library with PG pins.
- Multiple `operating_conditions` groups are defined, but the `default_operating_conditions` attribute is not defined in a library with PG pins.

The Library Compiler tool checks the `operating_conditions` group syntax and issues a warning message if any of the following conditions occur:

- Only one `operating_conditions` group is defined in a library with PG pins, a library with rails, or a library without PG pins or rails.
- Multiple `operating_conditions` groups and a `default_operating_conditions` attribute are defined in a library with PG pins, a library with rails, or a library without PG pins or rails.

On-Chip Variation Checks

The Library Compiler tool checks the syntax of an on-chip variation (OCV) model and issues an error message if any of the following rules are not met:

- Cross-reference rules
 - The `ocv_derate` group referenced by the `ocv_derate_group` or the `ocv_derate_distance_group` attributes must be specified in the `library` or the same `cell` group.
 - The `ocv_derate` group referenced by the `default_ocv_derate_group` or `default_ocv_derate_distance_group` attribute must be specified at the library level.
- Template reference rules
 - The `ocv_table_template` group that is referenced by the `ocv_derate_factors` group must be specified.
 - In the `ocv_table_template` group, the values of `variable_1` or `variable_2` must be only either `path_depth` or `path_distance`.
 - If the `ocv_table_template` group contains both `variable_1` and `variable_2`, the values of `variable_1` and `variable_2` must be different.
 - For parametric OCV library models, the `ocv_table_template` group must contain only `variable_1` with the value of `path_distance`.

- Data completion rules
 - If a library contains the `ocv_derate_distance_group` or `default_ocv_derate_distance_group` attribute, the library-level `distance_unit` attribute must be specified.
 - The `derate_type`, `rf_type`, and `path_type` attributes must be specified in the `ocv_derate_factors` group.

The Library Compiler tool checks the on-chip variation syntax and issues a warning message if any of the following conditions are not met:

- Multiple-groups rules
 - Each `ocv_derate` group must have a unique name.
If multiple `ocv_derate` groups have the same name, the group that is last specified overrides the previous ones.
 - The values of at least one of the `rf_type`, `derate_type`, and `path_type` attributes must be different in different `ocv_derate_factors` groups.
If multiple `ocv_derate_factors` groups have identical `rf_type`, `derate_type`, and `path_type` attribute values, the group that is last specified overrides the previous groups.
- Data completion rule
 - The `ocv_arc_depth` attribute must be defined in a library where the `default_ocv_derate_group` or `ocv_derate_group` attribute is specified.
If the `ocv_arc_depth` attribute is not defined in the library, the Library Compiler tool uses the default.

LVF Checks For Cell Delay, Transition, and Constraint

The Liberty variation format (LVF) syntax enables you to define the cell delay, output transition, and constraint variation at every input slew and load point in the delay, output transition, and constraint tables.

The Library Compiler tool checks the syntax of a parametric on-chip variation (OCV) Liberty variation format (LVF) model and issues an error message if any of the following rules are not met:

- Data completion rules
 - If the `ocv_sigma_cell_rise` or `ocv_sigma_cell_fall` groups are defined in a `timing` group, the corresponding NLDM delay, CCS timing, or compact CCS timing must be specified in that `timing` group.
 - If the `ocv_sigma_rise_constraint` or `ocv_sigma_fall_constraint` groups are defined in a `timing` group, the corresponding NLDM timing constraints must be specified in that `timing` group.
 - If the `ocv_sigma_rise_transition` or `ocv_sigma_fall_transition` groups are defined in a `timing` group, the corresponding NLDM transition groups must be specified in that `timing` group.
- Data exclusion rules
 - The `ocv_sigma_cell_rise` or `ocv_sigma_cell_fall` group must not be specified in a timing constraint arc.
 - The `ocv_sigma_rise_constraint` or `ocv_sigma_fall_constraint` group must not be specified in a delay arc.
 - The `sigma_type` attribute must not be specified in the `ocv_sigma_rise_constraint` or the `ocv_sigma_fall_constraint` group.
 - The `ocv_sigma_rise_transition` or `ocv_sigma_fall_transition` group must not be specified in a timing constraint arc.
 - The `ocv_sigma_rise_transition` and `ocv_sigma_fall_transition` groups must be separately defined for both the `early` and `late` values of the `sigma_type` attribute.

The Library Compiler tool checks the parametric OCV Liberty variation format (LVF) syntax and issues a warning message if the following condition is not met:

- Data completion rules
 - If the `ocv_sigma_cell_rise` or `ocv_sigma_cell_fall` groups are defined in a `timing` group, the table indexes must be identical to the indexes in the NLDM delay, CCS timing, or compact CCS timing specified in that `timing` group.

- If the `ocv_sigma_rise_constraint` or `ocv_sigma_fall_constraint` groups are defined in a `timing` group, the table indexes must be identical to the indexes in the NLDM timing constraint specified in that `timing` group.
- If the `ocv_sigma_rise_transition` or `ocv_sigma_fall_transition` groups are defined in a `timing` group, the table indexes must be identical to the indexes in the NLDM, CCS, or compact CCS timing transition specified in that `timing` group.
- The `sigma_type` attribute must be defined in both the `ocv_sigma_rise_transition` and the `ocv_sigma_fall_transition` groups.

If the `sigma_type` attribute is not defined in any of the `ocv_sigma_rise_transition` and the `ocv_sigma_fall_transition` groups, the Library Compiler tool assigns the default `early_and_late` value to the `sigma_type` attribute.

LVF-Moments OCV Checks

The Library Compiler tool checks the LVF moments-based syntax and issues an error message if any of the following rules are not met:

- Completeness check
 - If the `ocv_mean_shift_*` or `ocv_skewness_*` groups are defined in a `timing` group, the corresponding `ocv_std_dev_*` group must be defined in the same `timing` group.
 - If an `ocv_mean_shift_*` group is defined in a `timing` group, the corresponding `ocv_skewness_*` group must be defined in the same `timing` group and vice versa.
- Exclusiveness check
 - The `ocv_mean_shift_cell_rise`, `ocv_mean_shift_cell_fall`, `ocv_mean_shift_rise_transition`, `ocv_mean_shift_fall_transition` and the `ocv_skewness_cell_rise`, `ocv_skewness_cell_fall`, `ocv_skewness_rise_transition`, `ocv_skewness_fall_transition` groups must not be defined in a constraint arc.
 - The `ocv_mean_shift_rise_constraint`, `ocv_mean_shift_fall_constraint`, and the `ocv_skewness_rise_constraint`, `ocv_skewness_fall_constraint` groups must not be defined in a delay arc.
- Validity check

The values of the `ocv_std_dev_*` table must be greater than 0.0.

The tool checks the LVF moments-based syntax with the following rules and reports warning messages if they are not met:

- Consistency check

If the `ocv_mean_shift_*` or the `ocv_skewness_*` groups are defined in a `timing` group, the table indexes must be identical to the corresponding `ocv_std_dev_*` group table indexes defined in the same `timing` group.

- The `ocv_mean_shift_*` table values must be consistent with the `ocv_skewness_*` table values when:

$$\text{mean_shift} \times \text{skewness} < 0$$

- The `ocv_skewness_*` table values must be consistent with the `ocv_sigma_*` table values for both the `early` and `late sigma_type` attribute values when:

$$\text{skewness} \times (\sigma_{\text{late}} - \sigma_{\text{early}}) < 0$$

- The `ocv_std_dev_*` table values must be consistent with the `ocv_sigma_*` table values for both the `early` and `late sigma_type` attribute values when:

$$(\text{standard_deviation} - \sigma_{\text{early}}) \times (\text{standard_deviation} - \sigma_{\text{late}}) > 0$$

Retain Arc OCV Checks

The Library Compiler tool checks the syntax of an LVF retain arc model and issues an error message if any of the following rules are not met:

- If the `ocv_sigma_retaining_rise`, `ocv_sigma_retaining_fall`, `ocv_sigma_retain_rise_slew`, or `ocv_sigma_retain_fall_slew` group is defined in a `timing` group, the corresponding nominal retain arc table must also be defined in the same `timing` group.
- The `ocv_sigma_retaining_rise`, `ocv_sigma_retaining_fall`, `ocv_sigma_retain_rise_slew`, and `ocv_sigma_retain_fall_slew` groups must not be defined inside a constraint arc.
- If the `sigma_type` attribute is set to `early` for an `ocv_sigma_*` group, then the `sigma_type` attribute must be set to `late` for another `ocv_sigma_*` group of the same type.

The Library Compiler tool checks the syntax of an LVF retain arc model and issues a warning message if any of the following rules are not met:

- If the `ocv_sigma_retaining_rise`, `ocv_sigma_retaining_fall`, `ocv_sigma_retain_rise_slew`, or `ocv_sigma_retain_fall_slew` group is defined in a `timing` group, the table indexes must be identical to the indexes of the corresponding nominal retain arc table defined in the same `timing` group.
- If the `sigma_type` attribute is not specified in an LVF retain arc group, the default value of `early_and_late` is considered.

Characterization Configuration Checks

The Library Compiler tool checks the `char_config` group syntax and issues an error message if any of the following conditions occur:

- The `driver_waveform` attribute is defined within and outside the `char_config` group.
- The `unrelated_output_net_capacitance` attribute is defined in pin-groups that are not output pins.
- The `unrelated_output_net_capacitance` attribute is not defined, except for the `nldm_delay` and `nlpn_output` characterization models.
- The `default_value_selection_method` attribute has used the `follow_delay` method for a characterization model other than the `nldm_transition` model.
- Either the `default_value_selection_method` attribute, or the `default_value_selection_method_rise` and `default_value_selection_method_fall` attributes are not defined.
- The `merge_selection` attribute is not defined, when either the `merge_tolerance_abs` attribute or the `merge_tolerance_rel` attribute is defined.
- The `char_config` group is defined for a library that includes a three-state cell, but the `three_state_disable_measurement_method` attribute is not defined at the library level.
- The `three_state_disable_measurement_method` attribute is defined as `current`, but the `three_state_disable_current_threshold` attribute is not defined at the library level.
- The `three_state_disable_measurement_method` attribute is defined as `voltage`, but the `three_state_disable_monitor_node` attribute is not defined at the library level.
- The `char_config` group is defined for a library that includes a three-state cell, but the `three_state_cap_add_to_load_index` attribute is not defined at the library level.
- The `char_config` group is defined, but the `ccs_timing_segment_voltage_tolerance_rel` is not defined at the library level.
- The `char_config` group is defined, but the `ccs_timing_delay_tolerance_rel` attribute is not defined at the library level.
- The `char_config` group is defined, but the `ccs_timing_voltage_margin_tolerance_rel` attribute is not defined at the library level.
- The `char_config` group is defined, but the CCS receiver capacitance attributes are not defined at the library level.

- The values of the CCS receiver capacitance attributes are greater than 100.0 or less than 0.0.
- The `char_config` group is defined, but the input capacitance attributes are not defined at the library level.
- The values of the input capacitance attributes are greater than 100.0 or less than 0.0.

The Library Compiler tool checks the `char_config` group syntax and issues a warning message if any of the following conditions occur:

- When the `internal_power` group exists in a library, the library-level `char_config` group needs to be defined with the `internal_power_calculation` attribute.
- The `driver_waveform` attribute is not defined with the `all` characterization model in the library.
- The `driver_waveform` attribute is defined at the library level, and the `normalized_driver_waveform` group is defined without the `driver_waveform_name` attribute. In this case, the `normalized_driver_waveform` group is ignored, and not stored in the `.db` file.

Driver Waveform Checks

The Library Compiler tool checks the `normalized_driver_waveform` table data at the library level and cell level as described in the following sections.

Library-Level Checks

The Library Compiler tool checks the `normalized_driver_waveform` table data at the library level and issues a warning or error message if any of the following conditions occur:

- If multiple driver waveform tables are defined and the `driver_waveform_name` values are the same. In this case, only the last name is retained and the Library Compiler tool issues a warning message. Other tables with the same driver waveform name are ignored and are not stored in the `.db` file.
- If normalized voltage values in `index_2` are not monotonically increasing in the range of [0, 1]. If any value goes beyond this range, the Library Compiler tool issues an error message. If there are not enough start and endpoints in the waveform (at least one point between 0 ~ 0.05 and 0.95 ~ 1.0), the Library Compiler tool issues a warning message.
- If the time values for a specific input slew are not monotonically increasing and not overlapping.

Cell-Level Checks

The Library Compiler tool checks the `normalized_driver_waveform` table data at the cell level and issues a warning message if any of the following conditions occur:

- If there are not enough start and endpoints in the waveform (at least one point between 0 ~ 0.05 and 0.95 ~ 1.0).
- More than one common driver waveform table (a table without a `driver_waveform_name`) is defined in the library. In this case, only the last table is retained. The other tables are ignored and not stored in the .db file.
- More than one `normalized_driver_waveform` table with the same name is defined. In this case, only the last table is retained. The other tables are ignored and not stored in the .db file.
- If multiple `normalized_driver_waveform` templates are specified without the `driver_waveform_name` attribute to identify the `normalized_driver_waveform` template.

The Library Compiler tool checks the `normalized_driver_waveform` table data at the cell level and issues an error message if the following conditions occur:

- The `normalized_driver_waveform` table uses a nonexistent `lu_table_template` or uses the wrong `lu_table_template` where the `variable_1` and `variable_2` values are not equal to the `input_net_transition` and `normalized_voltage`.
- The `index_2` values (the normalized voltage) are not in monotonically increasing order, in the range of [0, 1].
- Any `index_2` value goes beyond the range of [0, 1].
- For a specific input slew, if the time values are not in monotonically increasing order or there is overlapping between the time values.
- The `driver_waveform_name` referred by the `driver_waveform`, `driver_waveform_rise`, and `driver_waveform_fall` attributes is not predefined.

Timing Arc Sensitization Checks

The Library Compiler tool checks sensitization information and reports warnings and errors as described in the following sections.

Sensitization Group Checks

The Library Compiler tool issues an error message if any of the following conditions occur in the `sensitization` group:

- No `pin_names` attribute is defined.
- The `pin_names` attribute is not declared first in the group.
- There is a duplicate pin name in the `pin_names` attribute.
- No vector attribute is defined.
- The `vector_id` value in the `vector` attribute is less than zero.
- The `vector_id` value in the `vector` attribute is duplicated in the sensitization group.
- The number of elements in `vector_string` is different from the number of arguments in the `pin_names` attribute.

Cell-Level Checks

The Library Compiler tool issues an error message if any of the following conditions occur in a `cell` group specified with sensitization information:

- The number of pins in the `pin_name_map` attribute is different from the number of arguments in the sensitization master's pin names.
- The pin name in the `pin_name_map` attribute is an undefined pin name in the current cell.
- There is a duplicate pin name in the `pin_name_map` attribute.
- If only the `sensitization_master` attribute is defined, but no `pin_name_map` attribute is defined, and the pin name in the `pin_names` attribute is undefined in the current cell.

Timing-Level Checks

The Library Compiler tool issues an error message if any of the following conditions occur when reading the `timing` group in a cell with sensitization information:

- There are `wave_rise` and `wave_fall` attributes found in the timing group but no `sensitization_master` attribute is defined either in the timing arc or in the cell.
- The pin name for `wave_rise` and `wave_fall` is undefined in the current cell. The priority for getting pin names for `wave_rise` and `wave_fall` in the timing arc is determined by the following order:
 1. The `pin_name_map` attribute in the current timing arc.
 2. The pin names of the `sensitization_master` attribute in the current timing arc.

3. The `pin_name_map` definition in the current cell.
 4. The pin names of the `sensitization_master` attribute in the current cell.
- The `pin_name_map` attribute is defined in the timing group and the number of pins in the attribute is different from that in the sensitization master. When you select a sensitization master group reference, higher priority is given to the `sensitization_master` group defined under the current timing group rather than the `sensitization_master` group specified at the cell level.
 - Any vector ID in `wave_rise` and `wave_fall` is not defined in the master sensitization group.
 - The vector ID number in `wave_rise` and `wave_fall` is less than 2.
 - The `wave_rise_sampling_index` and `wave_fall_sampling_index` value is less than zero or greater than or equal to the `wave_rise` and `wave_fall` size.
 - The number of arguments in `wave_rise_timing_interval` and `wave_fall_timing_interval` is less than 1 or greater than or equal to the `wave_rise` and `wave_fall` size.

Phase-Locked Loop Checks

The Library Compiler tool issues an error message if any of the following conditions are not met:

- If the pin-level `is_pll_reference_pin` attribute is specified, it must be set to `true` on an input pin.
- If the pin-level `is_pll_feedback_pin` attribute is specified, it must be set to `true` on an input pin.
- If the pin-level `is_pll_output_pin` attribute is specified, it must be set to `true` on an output pin.
- Only one of the following attributes at the most can be set to `true` in a pin group: `is_pll_reference_pin`, `is_pll_feedback_pin` and `is_pll_output_pin`.
- If the cell-level `is_pll_cell` attribute is set to `true`, the cell group must contain only one phase-locked loop reference pin and only one phase-locked loop feedback pin. However, it can contain one or more phase-locked loop output pins.

If the cell-level `is_pll_cell` attribute is not set to `true`, the pin-level `is_pll_reference_pin`, `is_pll_feedback_pin` and `is_pll_output_pin` attributes are ignored and the Library Compiler tool issues a warning message.

Physical Variant Cell Checks

The Library Compiler tool checks the syntax of physical variant cells and issues an error message if any of the following conditions are not met:

- Any of the cells listed by the `physical_variant_cells` attribute must not be same as the cells defined by the `cell` group.
- All the cell name strings listed under the `physical_variant_cells` attribute must be unique.

Decoupling Capacitor, Filler, and Tap Cell Checks

The Library Compiler tool performs the following checks on decoupling capacitor, filler, and tap cells. If the conditions are not met, the Library Compiler tool issues an error message.

- Only one of the following attributes can be set to true in a given cell: `is_decap_cell`, `is_filler_cell`, `is_tap_cell`
- Tap cells must have power and ground pins only.

Programmable Driver Type Checks

The Library Compiler tool checks the programmable driver type syntax and quits with an error if any of the following conditions are not met:

- The function string of any programmable function must be mutually exclusive within a pin.
- The function string of any programmable function must only contain input or inout pins.
- Either a nonprogrammable driver type using the old `driver_type` attribute or a programmable driver type function can be specified, but not both.
- The cell where the programmable driver type functions are defined must be a pad cell.
- The pin on which `pull_up_function` or `pull_down_function` is defined must be an input, output, or inout pin.
- The pin on which a `bus_hold_function` is defined must be an inout pin.
- The pin on which `open_drain_function`, `open_source_function`, `resistive_function`, `resistive_0_function`, or `resistive_1_function` is defined must be an output or inout pin.

Statetable Checks

In addition to checking for syntax errors, the Library Compiler tool checks for the following errors in `statetable` groups:

- An input that is not functionally required by any of the internal nodes
- An internal node that is purely combinational (that is, the node has no N)
- L/H or H/L present in the output without any L/H or H/L in the inputs
- An input or an internal node name in the statetable that is the same as another input or internal node name in the statetable
- Overlapping entries and unspecified entries, which cause warnings

In `pin` groups, the Library Compiler tool checks for the following errors:

- A functionally related input that is explicitly defined as don't care (-) in the `input_map` attribute
- Too many names defined in the `input_map` attribute
- An internal node name in the `input_map` attribute that does not correspond to the current port name, which results in a warning
- An `internal_node` attribute that does not match any statetable internal node name
- Ports that have missing or illegal attributes
- An input port defined as an internal node in the `input_map` attribute
- Specified internal pins that do not affect any outputs, which results in a warning
- Inclusion of illegal ports in the `state_function` expression; illegal ports are outputs without an `internal_node` attribute and inout without `three_state` attributes
- Not having any output with an `internal_node` attribute

Scan Cell Checks

The Library Compiler tool checks scan cells as described in the following sections.

test_cell Group Checks

The Library Compiler tool checks the `pin` groups in a `test_cell` group of a scan cell, and reports an error if the following rules are not met:

- Each `pin` in the `cell` group has a corresponding `pin` with the same name in the `test_cell` group.
- The `pin` group only includes the `direction`, `function`, `test_output_only`, and `signal_type` attributes. The `pin` group does not include timing, capacitance, fanout, or load information.
- Input pins either have a `signal_type` attribute or are referenced in an `ff`, `ff_bank`, `latch`, or `latch_bank` group.
- An output pin has either a `function` attribute or a `signal_type` attribute, or both.

Scan Cell Timing Arc Checks

The Library Compiler tool checks the timing arc consistency for the clock or enable pins, that have the `signal_type` attribute, in the `test_cell` group of the scan cells, and issues an error message if the following conditions are not met:

- For a `latch` group with an active-high enable pin, the corresponding timing arcs are modeled only by the `setup_falling`, `hold_falling`, `recovery_falling`, `removal_falling`, `skew_falling`, and `rising_edge` values of the `timing_type` attribute. Specifically,
 - The timing arc, from the enable pin to the data pin, is modeled by the `setup_falling` and `hold_falling` values.
 - The timing arc, from the enable pin to the output pin, is modeled by the `rising_edge` value.
- For a `latch` group with an active-low enable pin, the corresponding timing arcs are modeled only by the `setup_rising`, `hold_rising`, `recovery_rising`, `removal_rising`, `skew_rising`, and `falling_edge` values of the `timing_type` attribute. Specifically,
 - The timing arc, from the enable pin to the data pin, is modeled by the `setup_rising` and `hold_rising` values.
 - The timing arc, from the enable pin to the output pin, is modeled by the `falling_edge` value.

- For a `ff` group with an active-high clock pin, the corresponding timing arcs are modeled only by the `setup_rising`, `hold_rising`, `recovery_rising`, `removal_rising`, `skew_rising`, and `rising_edge` values of the `timing_type` attribute. Specifically,
 - The timing arc, from the clock pin to the data pin, is modeled by the `setup_rising` and `hold_rising` values.
 - The timing arc, from the clock pin to the output pin, is modeled by the `rising_edge` value.
- For a `ff` group with an active-low clock pin, the corresponding timing arcs are modeled only by the `setup_falling`, `hold_falling`, `recovery_falling`, `removal_falling`, `skew_falling`, and `falling_edge` values of the `timing_type` attribute. Specifically,
 - The timing arc, from the clock pin to the data pin, is modeled by the `setup_falling` and `hold_falling` values.
 - The timing arc, from the clock pin to the output pin, is modeled by the `falling_edge` value.

Multibit Scan Cell Checks

The Library Compiler tool checks multibit scan cells, and reports an error for the following conditions:

- A multibit scan cell does not have the bundle or bus output pins.
- The `function` attribute is on a single-bit output pin of the multibit scan cell.

Checks for Multibit Scan Cells With Internal Scan Chains

The Library Compiler tool checks the multibit scan cells with internal scan chains, and reports errors if the following conditions are not met:

- The `scan_start_pin` attribute must be specified in a `bus` or `bundle` group where the `direction` attribute value is either `inout` or `output`.
- The `scan_start_pin` attribute must be specified in a cell where a single-bit pin (not part of a `bus` or `bundle`) has the `signal_type` attribute set to `test_scan_in`, `test_scan_out`, or `test_scan_out_inverted`.
- The `single_bit_degenerate` attribute must be specified in a cell where the `bus` or `bundle` group includes the `scan_start_pin` attribute.
- The pin specified in the `scan_start_pin` attribute must be a pin of the `bus` or `bundle` group. The pin must be the first or last bit of the `bus` or `bundle`.
- The `scan_start_pin` attribute must be specified in the same `bus` or `bundle` group where the `scan_pin_inverted` attribute is specified.

Scan-Enabled LSSD Cell Checks

The Library Compiler tool checks various types of scan-enabled LSSD cells, and reports errors when the following conditions are not met:

- The output pin timing arcs related to the `test_scan_clock_b` signal-type, are grouped to the slave latch.
- The timing arcs related to the clock or enable pins other than the output pin, that is, the `test_scan_clock_a`, `test_scan_clock_b`, `test_scan_clock`, and `test_clock` signal types, are grouped to the master latch.
- Timing arc consistency is checked on the master- and slave-latch groups separately.

Generic Integrated Clock-Gating Cell Checks

The Library Compiler tool checks the syntax of generic integrated clock-gating cells, and issues an error message if the following rules are not met:

- The `clock_gating_integrated_cell` attribute must not be set. The Library Compiler tool infers the clock-gating structure of the cell based on the values of the `latch` group and the `function` attribute.
- The `pin_is_active_high` attribute must not be set.
- The `clock_gate_reset_pin` attribute must not be set for the `latch_posedge_postcontrol` generic integrated clock-gating cell. The Library Compiler tool infers these values based on the pins.

The Library Compiler tool checks the syntax of generic integrated clock-gating cells, and issues a warning when the following conditions occur:

- The generic integrated clock-gating cell is not of a specific type.
- The generic integrated clock-gating cell includes a flip-flop.

Advanced Low-Power Screener Checks

The following sections describe the types of checks that the `read_lib` command performs for low-power library cells:

- [PG Pin Library Checks](#)
- [Partial PG Pin Cell Checks](#)
- [Feedthrough Cell Checks](#)
- [Overdrive and Underdrive Voltage Checks](#)
- [Unconnected Pin Checks](#)
- [Level-Shifter Cell Checks](#)
- [Isolation Cell Checks](#)
- [Clamp Function Checks](#)
- [Internal Isolation Checks for Macro and Pad Cells](#)
- [Switch Cell Checks](#)
- [Retention Cell Checks](#)
- [Always-On Cell Checks](#)
- [Antenna-Diode Cell Checks](#)

PG Pin Library Checks

The Library Compiler tool issues an error message if the following conditions occur when you are modeling a standard or macro cell using the power and ground pin syntax:

- A `pg_pin` group does not have the `voltage_name` attribute. This error is not generated for a level-shifter cell not powered by the switching domains.
- If the `voltage_map` complex attribute is defined at the library level and all cells in the library do not have the `pg_pin` groups defined for all cells.
- If the `voltage_map` complex attribute is specified inside the operating conditions group.
- If you do not have at least one power pin and one ground pin in a cell modeled using the power and ground pin syntax.
- If more than one `user_pg_type` name is specified for a power and ground pin.
- If `physical_connection` is only associated with the power and ground pins of the following predefined bias types: `pwell`, `nwell`, `deepnwell`, `deeppwell`.

- If the `related_bias_pin` attribute value in the `pg_pin` and signal pin group is not a valid `pg_pin` power and ground pin type.
- If the `pg_pin` power type is not correctly associated with a `nwell` bias power and ground pin and the `pg_pin` ground type is not correctly associated with a `pwell` bias power and ground pin.

Note:

This rule does not apply to silicon-on-insulator (SOI) cells.

- If the association between the power and ground pin and the bias pin in the signal pin is not correct.
- The `output_signal_level_low` and `output_signal_level_high` attributes are not defined as a pair.
- The value of the `output_signal_level_high` attribute is less than the `output_signal_level_low` attribute value.

The Library Compiler tool issues a warning message if the following conditions occur when you are modeling a standard cell using the power and ground pin syntax:

- If only one `pwell` or `nwell` bias pin is added to any functional cell.

Bias PG Pin Checks

The Library Compiler tool checks the syntax of the insulated bias PG pins with the following rules, and reports error messages if they are not met.

- The `is_insulated` and `tied_to` attributes must be defined only in a bias `pg_pin` group, that is, a `pg_pin` group where the `pg_type` attribute is set to `pwell`, `nwell`, `deepnpwell`, or `deepnwell`.
- When both the `is_insulated` and the `tied_to` attributes are defined in a bias `pg_pin` group, the PG pin direction must be internal.
- When both the `is_insulated` and the `tied_to` attributes are defined in a bias `pg_pin` group, the `physical_connection` attribute must not be defined.
- The `voltage_map` attribute value of the insulated bias PG pin must match the `voltage_map` attribute value of the PG pin it is tied to.

When the `is_insulated` attribute is defined and the `tied_to` attribute is not defined in an insulated bias PG pin, the Library Compiler tool generates a warning message and the cell is marked with the `dont_use` and `dont_touch` attributes.

Partial PG Pin Cell Checks

With the exception of ETM cells, the Library Compiler tool checks cells with partial PG pins and issues an error message if the following conditions are not met.

- A cell must comply with the primary PG requirement principle, meaning that a cell is treated as a violation if it contains the following information but does not have at least one power and one ground PG specified in the library:
 - Functional cell modeling information, including `function` or `state_function` in an `ff`, `ff_bank`, `latch`, `latch_bank`, or a `statetable` group to model the cells function. (Tied-off cells do not need to meet this requirement.)
 - Power data, including cell leakage power information defined in the `cell_leakage_power` attribute and the `leakage_power` group; internal power in the `pin` group; or CCS power models, including leakage current tables and dynamic current tables.
 - Noise data, including any cell noise information.
 - Timing data, including any cell timing information.
 - Macro and analog IP cell information, including cells that specify the `is_macro_cell` attribute or the `power_cell_type` attribute set to `macro`.

Feedthrough Cell Checks

The Library Compiler tool checks the syntax of multipin feedthroughs, and issues error messages if the following rules are not met:

- The `short` attribute pin-name list must include at least two pin names.
- The `related_power_pin` or the `related_ground_pin` attribute must not be specified for feedthrough pins listed using the `short` attribute.
- The ports or pins listed in the `short` attribute must have identical `capacitance`, `rise_capacitance`, or `fall_capacitance` values.

Overdrive and Underdrive Voltage Checks

The Library Compiler tool checks the syntax of signal pins with overdrive and underdrive voltages, and issues error messages if the following rules are not met:

- The values of the `max_input_delta_overdrive_high` and `max_input_delta_underdrive_high` attributes must be constant and non-negative.
- The `max_input_delta_overdrive_high` and `max_input_delta_underdrive_high` attributes must be specified only for input and inout signal pins of macro cells, memory cells, switch cells, level-shifter cells, and pad cells.

- The rail voltage of a level-shifter cell that does not have the `input_signal_level` attribute must be within the voltage range modeled by the `max_input_delta_overdrive_high` and `max_input_delta_underdrive_high` attributes.

The Library Compiler tool checks the syntax of signal pins with overdrive and underdrive voltages, and issues warning messages if the following rules are not met:

- The voltage range modeled by the `max_input_delta_overdrive_high` and `max_input_delta_underdrive_high` attributes must be within the `input_voltage_range` values of the pin.

Unconnected Pin Checks

The Library Compiler tool checks the syntax of internally unconnected pins, and issues error messages if the following rules are not met:

- The `is_unconnected` attribute must be set to `true` for a pin, bus, or bundle that is not part of the `short` attribute syntax and does not have a `related_power_pin` or `related_ground_pin` attribute.

Level-Shifter Cell Checks

The Library Compiler tool checks the conditions of level-shifter cells and issues an error message if any of the following conditions are not met:

- The data input and output pins cannot be related to the same power pin.
- The `level_shifter_type`, `input_voltage_range`, `output_voltage_range`, and `std_cell_main_rail` attributes must be specified only in a level-shifter cell that has the `is_level_shifter` attribute set to `true`.
- The `std_cell_main_rail` attribute must satisfy the following properties:
 - The attribute must be specified as the `related_power_pin` value of one of the level-shifter signal pins. This rule does not apply to a level-shifter cell that is not powered by the switching domains.
 - The attribute must be defined in a `primary_power` power and ground pin.
- The `input_voltage_range` and `output_voltage_range` attributes should always be defined together.
- The `input_voltage_range` and `output_voltage_range` `lower_bound` value must be less than or equal to the `upper_bound` value, as shown:


```
input_voltage_range (1.0, 2.0);
```

- The `level_shifter_data_pin` pin-level attribute can appear only in a level shifter indicated by the `is_level_shifter` attribute.
- A pin with the `alive_during_power_up` attribute must also have the `isolation_cell_data_pin` or the `level_shifter_data_pin` attribute set to `true`.
- The `input_voltage_range` and `output_voltage_range` values must be consistent with the type of level shifter.

In the following example,

```
cell level_shifter_cell_name) {
    is_level_shifter : true;
    level_shifter_type : level_shifter_type_value;

    /* input_voltage_range and output_voltage_range can be
    specified at the pin level, the cell level, or in both places */

    input_voltage_range (float11min, float12max);
    output_voltage_range( float21min, float22max);
    ...
    /* The minimum lower bound for the input_voltage_range
    attributes for all input pins, and for the cell itself, is float11,
    and the maximum upper bound is float12 */

    pin(input_pin_name) {
        direction : input;
        input_voltage_range ( float11 , float12);
        ...
    }
    /* The minimum lower bound for the output_voltage_range
    attributes for all output pins, and the cell itself, is float21, and
    the maximum upper bound is float22 */

    pin(output_pin_name) {
        direction : output;
        output_voltage_range (float21, float22);
        ...
    }
}
```

The tool must ensure that the following constraints are met:

- If the `level_shifter_type` value is HL, then `float12` cannot be less than or equal to `float21`.
- If the `level_shifter_type` value is LH, then `float11` cannot be greater than or equal to `float22`.
- The value specified in `float11` cannot be greater than `float12`.
- The value specified in `float21` cannot be greater than `float22`.

- The `level_shifter_enable_pin` pin-level attribute can be defined only on an input pin.
- The number of input pins must be equal to the number of output pins.
- If enable pins (with the `level_shifter_enable_pin` attribute) exist, the number of enable pins must also be equal to the number of output pins.
- Each of the input pins should define the `input_signal_level` value or the `related_power_pin` and `related_ground_pin` as a pair, or both. Similarly, the output pins should define the `related_power_pin` and `related_ground_pin` value as a pair.
- The `input_signal_level` value of the data input pin and `output_signal_level` value of the output pin should be different.

This also applies if the `related_power_pin` and the `related_ground_pin` attributes only have been specified. In this case, the `related_power_pin` value of the input side cannot be equal to the `related_power_pin` value of the output pin.

The Library Compiler tool does not check or issue error messages regarding the following:

- The `input_signal_level` value of the enable input pin against the `input_signal_level` value of the data input pin, the `output_signal_level` value of the output pin, or the `related_power_pin` and `related_ground_pin` pairs that are specified on the input and the output pin.
- The power and ground pin associations of the enable input pin against the power and ground pin associations of the data input pin or the power and ground pin associations of the output pin.
- The cell-level `input_voltage_range` and `output_voltage_range` attributes and pin-level `input_voltage_range` and `output_voltage_range` attributes are mutually exclusive. In other words, you can specify the `input_voltage_range` and `output_voltage_range` attributes both at the cell level and at the pin level. If you do this, the attributes at the pin level have higher priority over those specified at the cell level.

Note:

The cell-level and pin-level `input_voltage_range` and `output_voltage_range` attributes are optional in any level-shifter cell.

Isolation Cell Checks

The Library Compiler tool checks the conditions of isolation cells and issues an error message if any of the following conditions are not met.

- The `isolation_cell_data_pin` and `isolation_cell_enable_pin` pin-level attributes can only appear in an isolation cell identified by the cell-level `is_isolation_cell` simple attribute.

- The `isolation_cell_data_pin` and `isolation_cell_data_pin` pin-level attributes can only be specified on an input pin.
- A pin with the `alive_during_power_up` attribute must also have the `isolation_cell_data_pin` or the `level_shifter_data_pin` attribute set to `true`.
- An isolation cell must have at least one enable pin and one data pin.
- The `related_power_pin` and `related_ground_pin` information should be specified on the input and output pins.

Note:

The Library Compiler tool issues an error message if the values of the power pin's associations on the input data pin and output pin are different from each other.

The Library Compiler tool does not check or issue error messages regarding the following:

- The power and ground pin associations of the enable input pin against the power and ground pin associations of the data input pin or the power and ground pin associations of the output pin.
- If both the `is_level_shifter` and `is_isolation_cell` attributes (set to `true`) are specified on the cell.

The Library Compiler tool checks the conditions of isolation cells with partial power down modeling and issues an error message if any of the following conditions are not met.

- All the following partial power down conditions must be true:
 - The `alive_during_partial_power_down` attribute must be set to `true` on the input pin whose related PG pin (with `isolation_cell_enable_pin: true`) has the `permit_power_down` attribute set to `true`.
 - The `alive_during_partial_power_down` attribute must be set to `true` on the output pin and the `permit_power_down` attribute on the related PG pin must also be set to `true`.
 - One or more power and ground pins must have the `permit_power_down` attribute set to `true`.
- An input pin can have the `alive_during_partial_power_down` attribute only when it also has the `isolation_cell_enable_pin` attribute.
- An output pin with the `alive_during_partial_power_down` attribute must have the corresponding `function` and `power_down_function` attributes.

The Library Compiler tool checks the conditions of clock-isolation cells and issues an error message if any of the following rules are not met.

- Only one input pin must have the `clock_isolation_cell_clock_pin` attribute set to `true`.

- Only one input pin must have the `isolation_cell_enable_pin` attribute set to `true`.
- An input pin cannot have both the `isolation_cell_enable_pin` and `clock_isolation_cell_clock_pin` attributes set to `true`.

Clamp Function Checks

The Library Compiler tool checks the clamp function attributes (`clamp_0_function`, `clamp_1_function`, `clamp_z_function`, `clamp_latch_function`, and `illegal_clamp_condition`) and issues an error message if any of the following conditions are not met.

- The clamp function attributes must be specified only in the output and inout pins, and not in the input pins.
- The Boolean conditions of the clamp function attributes must be mutually exclusive.
- The clamp function attributes must be functionally consistent with the output pins where they are defined.
- The Boolean condition of the clamp function attributes must include the pin with the `level_shifter_enable_pin` attribute (for enable level-shifter cells) or the `isolation_cell_enable_pin` attribute (for isolation cells).

Internal Isolation Checks for Macro and Pad Cells

The Library Compiler tool checks the following conditions for cells with isolation and issues an error message if they are not met:

- The `is_isolated` attribute is used on pins, buses, and bundles of macro cells and pad cells to identify internally isolated pins. The `is_isolated` attribute cannot be defined for pins, buses, and bundles of other type of cells.
- The `is_isolated` attribute cannot be defined in the pin, bus, or bundle group where the `always_on` attribute is already defined. This is because the `always_on` pins cannot be floating pins.
- The Boolean expression of the `isolation_enable_condition` attribute includes pins, buses, or bundles that have the `is_isolated` attribute.

The Library Compiler tool checks the following conditions for cells with internal isolation and issues a warning for the following conditions:

- The `isolation_enable_condition` attribute is defined on pin, bus, or bundle groups. However, the `is_isolated` attribute is not set to `true` for these groups. The Library Compiler tool issues a warning and automatically sets the `is_isolated` attribute to `true` for these groups.

- The `isolation_enable_condition` attribute is defined on the output pin group. The Library Compiler tool issues a warning that the `isolation_enable_condition` attribute is ignored by Synopsys tools.

The Library Compiler tool checks the following conditions for cells with internal PG pins, and issues an error message if they are not met:

- The `pg_function` attribute must be specified on the internal PG pins of only a macro and fine-grain switch cell.
- Each internal PG pin of a cell must have the `pg_function` attribute.

Note:

To enable this check, set the `lc_derived_pgpin_must_have_pg_function` variable as follows:

```
lc_shell> set lc_derived_pgpin_must_have_pg_function true
```

The default is `false`.

- The `direction` attribute must not be set to `input` on internal PG pins of a cell.

Switch Cell Checks

The Library Compiler tool checks switch cells and issues an error message if any of the following conditions are not met. The `read_lib` command sets the switch cell with the `dont_use` attribute.

- The switch function can contain only switch pins whose input pin (identified by the `switch_pin` attribute) is set to `true`. (The `function` attribute can also contain switch pins for a coarse-grain switch cell.)

Note:

The Library Compiler tool does not check whether timing is modeled by the composite current source (CCS) timing or the nonlinear delay model (NLDM) timing, and whether power is modeled by the CCS or NLDM power.

- You can specify the `switch_function` and `pg_function` attributes of PG pins only when the `pg_type` is either `internal_power` or `internal_ground`.
- The Boolean expression of the `switch_function` attribute in a fine-grain or coarse-grain switch cell can include only input or inout signal pins.
- All the signal pins defined in the Boolean expression of the `switch_function` attribute should have the `switch_pin` attribute set to `true`.
- The `pg_type` attribute of PG pins in `pg_function` can only be one of the following values: `primary_power`, `primary_ground`, `backup_power`, or `backup_ground`.

Fine-Grain Switch Cell Checks

A fine-grain switch cell must satisfy the following rules:

- Fine-grain cells must have the `pg_function` attribute on the internal PG pin. The internal PG pin need not have the `switch_function` attribute.
- The Boolean expression of the `pg_function` attribute must not have only signal pins, that is, it must have at least one PG pin.

Coarse-Grain Switch Cell Checks

A coarse-grain switch cell must satisfy the following rules:

- The `switch_cell_type` attribute must be specified with the `coarse_grain` value.
- The `switch_function` and `pg_function` attributes must be defined on the output or inout PG pin.
- The `switch_function` attribute must be defined to identify the control logic of its switch pins.
- It must have at least one switch pin.
- It must have at least one controlled power and ground pin and one regular power and ground pin (that is, a virtual VSS power and ground pin and a VSS power and ground pin, or a virtual VDD power and ground pin and a VDD power and ground pin). Each of these output power pins must have a `pg_function` Boolean expression containing input power pins.
- The `pg_function` attribute must contain only the input power and ground PG pins.
- The `related_switch_pin` value can be either an internal pin or a switch pin. If the `switch_pin` attribute is set to true within a pin, it is identified as a switch pin of the cell. If the `direction` attribute is set to internal within a pin group, it is identified as an internal pin of the cell. It is permitted to have the `related_pin` value within an internal pin in a valid `switch_pin` of the cell.

The Library Compiler tool checks coarse-grain multithreshold-CMOS models and reports a warning message if any of the following conditions occur:

- A negative conductance from an I-V curve is calculated.
- The largest conductance value that is calculated across all input voltage and output voltage indexes is zero. If the value is zero, the Library Compiler tool changes the value to the default conductance value of 0.01.
- At any input voltage, as the output voltage increases, the conductance does not decrease monotonically.

Retention Cell Checks

The Library Compiler tool issues an error message if any of the following conditions are not met:

- The `retention_condition` group cannot be specified on a cell missing the cell-level `retention_cell` attribute, that is, the `retention_condition` group must be only specified for retention cells.
- The number of pins defined by the `reference_pin_names` variable must match the number of pins defined by the `reference_input` attribute; there is no don't care concept in this model.
- The pin name specified by the `reference_input` attribute must be a valid pin, either an input, internal, or inout pin.
- The node name specified by the `function` attribute must match either the `variable1` or the `variable2` variable in the `ff,latch`, `ff_bank`, or `latch_bank` group.
- The variable in the `ff`, `latch`, `ff_bank`, or `latch_bank` group must have a unique name within a cell.
- A `function` attribute must have only one flip-flop or latch variable if any of the `ff,latch`, `ff_bank`, or `latch_bank` groups includes a `reference_pin_names` variable.
- The primary power and ground pins of the retention cell must be specified when the `required_condition` attribute in the `retention_condition` group is defined.
- A cell that includes the `retention_pin` complex attribute specified on any signal pins of the cell must include the cell-level `retention_cell` attribute.
- The `retention_pin` attribute must be specified on an output pin.
- If a pin does not include the `retention_pin` attribute, the inclusion of any of the following attributes is not permitted:
 - `save_action`
 - `restore_action`
 - `save_condition`
 - `restore_condition`
 - `restore_edge_type`
- If a pin includes the `retention_pin` attribute and the `pin_class` value is `save`, the following attributes are not allowed:
 - `restore_action`
 - `restore_condition`

- `restore_edge_type`
- If a pin includes the `retention_pin` attribute and the `pin_class` value is `restore`, the following attributes are not allowed:
 - `save_action`
 - `save_condition`
- The `clock_condition`, `preset_condition`, and `clear_condition` groups cannot be specified if either the `retention_cell` attribute or a `ff` or `latch` group is missing.
- If a pin includes the `retention_pin` attribute and the `pin_class` value is either `save` or `save_restore`, the following attributes must be included:
 - `save_action`
 - `save_condition`
- If a pin includes the `retention_pin` attribute and the `pin_class` value is either `restore` or `save_restore`, the following attributes must be included:
 - `restore_action`
 - `restore_condition`
 - `restore_edge_type`
- If the `restore_edge_type` attribute is required because the `pin_class` value is either `restore` or `save_restore`, but it is not specified, it defaults to `trailing`. See [“Mapping Rules to Disable the State Value of the retention_pin Attribute” on page 1-34](#) for the default values for missing `save_action` and `restore_action` attributes.
- Consistency checks are performed for the `save_action`, `restore_action`, and `retention_pin` attributes, as described in [“Mapping Rules to Disable the State Value of the retention_pin Attribute” on page 1-34](#).

Note:

The `retention_pin` and `power_gating_cell` attributes cannot be specified in the same cell due to incompatibilities between the old and new syntax.

Mapping Rules to Disable the State Value of the retention_pin Attribute

Table 1-1 highlights the mapping rules to disable the state value for the `retention_pin` attribute for different `restore_action` and `restore_edge_type` attribute values.

Table 1-1 Mapping Rules for `restore_action`

<code>restore_action</code>	<code>restore_edge_type</code>	Disable value (in <code>retention_pin</code> attribute)
L	leading, trailing	1
H	leading, trailing	0
R	leading	0
F	leading	1
R	trailing	1
F	trailing	0
R/F	edge_trigger	Any (N/A)

Table 1-2 highlights the mapping rules to disable the state value for the `retention_pin` attribute for different `save_action` attribute values.

Table 1-2 Mapping Rules for `save_action`

<code>save_action</code>	Disable value (in <code>retention_pin</code> attribute)
L, R	1
H, F	0

The Library Compiler tool issues a warning message if any of the following conditions are not met:

- The Boolean expression of the `required_condition` attribute in the `retention_condition` group must only have pins marked with the `retention_pin` attribute or input signal pins powered by `backup_power` or `backup_ground` pins.
- For a two-pin retention cell, the save control pin logic in the `required_condition` attribute in the `retention_condition` group must be the disable value.

- For a single-pin retention cell, the save-restore control pin logic in the `required_condition` attribute in the `retention_condition` group must be the enable value.

Zero-Pin Retention Cell Checks

The tool checks the syntax of a zero-pin retention register cell and issues an error message if the following condition is not met:

- The `retention_pin` attribute must not be specified on any of the signal pins, in particular the clock pins.

The tool checks the syntax of a zero-pin retention register cell and issues warning messages if any of the following conditions are not met:

- The cell must have a live-slave latch.
- The cell must have a clock signal.
- The `retention_condition` group must be specified and the `required_condition` attribute must be specified in this group. Otherwise, the tool marks the cell as a black box.
- The pins specified in the Boolean expression of the `required_condition` attribute in the `retention_condition` group must be input signal pins powered by the `backup_power` or `backup_ground` pins.
- The clock pin logic in the Boolean expression of the `required_condition` attribute in the `retention_condition` group must be the disable value for the alive latch.
- The asynchronous preset or clear pin logic in the `required_condition` attribute in the `retention_condition` group must be the inactive value for the alive latch.

Multibit Retention Scan Cell Checks

The Library Compiler tool checks multibit retention scan cells and reports warning messages when the following rule is not met:

- The multibit retention scan cell must be modeled with only one pair of sequential (`ff`, `latch`) groups. For multibit pin mapping, the `reference_input` attribute must be used.

Always-On Cell Checks

The Library Compiler tool generates an error if a signal pin specified with the `always_on` attribute in an always-on cell is not related to a backup power supply type of power and ground pin. However, it is acceptable if the ground is a primary ground pin or a backup ground pin.

The Library Compiler tool generates an error if a cell is identified as always-on with the `always_on` attribute, but the pin's `related_power_pin` attribute does not specify a backup power pin or a backup ground pin.

Antenna-Diode Cell Checks

The Library Compiler tool checks the following rules for antenna-diode cells and issues errors if any of the rules are not met:

- If the `antenna_diode_type` attribute is specified at the cell level, the cell must have only one data pin with the `direction` attribute set to either `input` or `inout`.
- If the `antenna_diode_type` attribute is set to `power_and_ground` at both the cell and pin levels, the `related_power_pin` and `related_ground_pin` attributes must be specified for the data pin.
- If the `antenna_diode_type` attribute is set to `power` at both the cell and pin levels, the `related_power_pin` attribute must be specified for the data pin.
- If the `antenna_diode_type` attribute is set to `ground` at both the cell and pin levels, the `related_ground_pin` attribute must be specified for the data pin.
- If the `antenna_diode_type` attribute is specified only at the pin level, the cell must be a macro cell and the `direction` attribute of the data pin must be set to either `input` or `inout`.

Note:

If the `antenna_diode_type` attribute is not specified for the antenna-diode cell, the Library Compiler tool does not issue a warning or an error message.

Composite Current Source (CCS) Screener Checks

The following sections describe the types of checks that the `read_lib` command performs for CCS libraries:

- [CCS Driver Model Checks](#)
- [Two-Segment Receiver Capacitance Model Checks](#)
- [Current Polarity and Full Voltage Swing Checks](#)

CCS Driver Model Checks

The following rules apply to composite current source driver models:

- The variable lists in the `output_current_template` referred to in the `vector` group must include only `input_net_transition`, `total_output_net_capacitance`, and `time`.
- The `index_1`, `index_2`, and `index_3` variables are required in the lookup tables that reference `output_current_template`; however, they are optional in `output_current_template`.
- For half-unate sequential timing arcs,
 - If the `timing_type` attribute is specified as `rising_edge` and the `timing_sense` attribute is specified as `positive_unate`, only the `output_current_rise` group is required.
 - If the `timing_type` attribute is specified as `falling_edge` and the `timing_sense` attribute is specified as `positive_unate`, only the `output_current_fall` group is required.
 - If the `timing_type` attribute is specified as `rising_edge` and the `timing_sense` attribute is specified as `negative_unate`, only the `output_current_fall` group is required.
 - If the `timing_type` attribute is specified as `falling_edge` and the `timing_sense` attribute is specified as `negative_unate`, only the `output_current_rise` group is required.

For all timing arcs that are not half-unate, `output_current_rise` and `output_current_fall` must be defined in a pair. Otherwise, the Library Compiler tool issues an error message. It is permitted to define only `output_current_rise` or `output_current_fall` inside the `timing` group for half-unate arcs.

- The `output_current_rise` and `output_current_fall` groups can coexist with other delay tables in the `timing` group.

- If more than one `output_current_rise` or `output_current_fall` group is defined in a timing group, only the last group defined is used.
- Inside each `output_current_rise` or `output_current_fall` group, `reference_time` must be the same for each load of an `input_net_transition` value.
- In the `vector` group, the indexes for `input_net_transition` and `total_output_net_capacitance` can define only one value each. The Library Compiler tool issues an error message if `output_current_rise` and `output_current_fall` vectors do not fully fill the entire `input_net_transition` and `total_output_net_capacitance` two-dimensional table.

For every `input_net_transition` and `total_output_net_capacitance` pair, there must be a current time vector. The same value of load points should be specified for each slew. Otherwise, the Library Compiler tool issues an error message.

- For output or inout pins,
 - The maximum value of the `input_net_transition` index of each CCS driver capacitance model must be greater than the `max_transition` attribute value.
 - The maximum value of the `total_output_net_capacitance` index of each CCS driver capacitance model must be greater than the `max_capacitance` attribute value.

Otherwise, the Library Compiler tool generates a warning message.

Two-Segment Receiver Capacitance Model Checks

The `read_lib` command performs the following checks for the two-segment receiver capacitance models:

- The `char_when` attribute must be specified in a pin-level `receiver_capacitance` group that represents a default condition arc, that is, does not have the conditional `when` attribute. Otherwise, the tool generates an error message.
- For a timing arc with both the `char_when` and `when` attributes, the `char_when` attribute condition must be equal to or a subset of the `when` attribute condition. Otherwise, the tool generates a warning message.
- In the `receiver_capacitance` group, the `lu_table_template` group referred to by `receiver_capacitance1_rise`, `receiver_capacitance1_fall`, `receiver_capacitance2_rise`, and `receiver_capacitance2_fall` groups contains only `input_net_transition` as its parameter.
- If any of `receiver_capacitance1_rise`, `receiver_capacitance1_fall`, `receiver_capacitance2_rise`, or `receiver_capacitance2_fall` are defined in the `receiver_capacitance` group, then all of them have to be defined.

- If more than one `receiver_capacitance1_rise`, `receiver_capacitance1_fall`, `receiver_capacitance2_rise`, or `receiver_capacitance2_fall` are defined in the `receiver_capacitance` group, only the last one counts.
- The `receiver_capacitance` group applies only to input or inout pins.
- If input pin A defines the `receiver_capacitance` group, then no timing arc A->Y (attached under pin Y) can define `receiver_capacitance1_rise`, `receiver_capacitance1_fall`, `receiver_capacitance2_rise`, or `receiver_capacitance2_fall`.

The Library Compiler tool accepts arc-based and pin-based receiver capacitance models for an input pin. Certain type of cells can have both arc- and pin-based receiver capacitances characterized and represented in the library.

- The receiver capacitance must be specified for each switching and non-switching state in a `pin` or `timing` group of a composite current source (CCS) timing model for any cell—that is, you must specify all the receiver capacitance `when` attribute conditions. Otherwise, the `read_lib` command generates a warning message.
- For input or inout pins,
 - The maximum value of the `input_net_transition` index of each pin-based composite current source (CCS) receiver capacitance model must be greater than the `max_transition` attribute value.
 - The maximum value of the `total_output_net_capacitance` index of each pin-based CCS receiver capacitance model must be greater than the `max_capacitance` attribute value.

Otherwise, the Library Compiler tool generates a warning message.

The `read_lib` command performs the following checks for timing-level receiver capacitance groups:

- In the `timing` group, the `lu_table_template` referred to by `receiver_capacitance1_rise`, `receiver_capacitance1_fall`, `receiver_capacitance2_rise`, and `receiver_capacitance2_fall` must contain `input_net_transition` and `total_output_net_capacitance` as its parameters.
- If any of `receiver_capacitance1_rise`, `receiver_capacitance1_fall`, `receiver_capacitance2_rise`, or `receiver_capacitance2_fall` are defined in the `timing` group, then all of them have to be defined.
- If more than one `receiver_capacitance1_rise`, `receiver_capacitance1_fall`, `receiver_capacitance2_rise`, or `receiver_capacitance2_fall` are defined in the `timing` group, only the last one counts.

- For inout pins,
 - The maximum value of the `input_net_transition` index of each arc-based CCS receiver capacitance model must be greater than the `max_transition` attribute value.
 - For inout pins, the maximum value of the `total_output_net_capacitance` index of each arc-based CCS receiver capacitance model must be greater than the `max_capacitance` attribute value.

Otherwise, the Library Compiler tool generates a warning message.

Multisegment Receiver Capacitance Model Checks

If either the `receiver_trip_threshold_pct_rise` or the `receiver_trip_threshold_pct_fall` attribute is present in the `library` group, the `read_lib` command considers the library to be a multisegment receiver capacitance model and performs the following checks:

- Both the `receiver_trip_threshold_pct_rise` and `receiver_trip_threshold_pct_fall` attributes must be specified and must have the same number of floating-point values.
- The `receiver_trip_threshold_pct_rise` floating-point values must be in increasing order and the `receiver_trip_threshold_pct_fall` values must be in decreasing order. The values must be bounded by 0.0 and 100.0.
- In a `receiver_capacitance` or `timing` group, there must be N `receiver_capacitance_rise` and N `receiver_capacitance_fall` groups with exclusive `segment` attribute values from 1 to N. N is the number of segments equal to one less than the number of floating-point values of the `receiver_trip_threshold_pct_rise` or the `receiver_trip_threshold_pct_fall` attribute.

Missing, overlapping, or invalid `segment` attribute values result in error messages.

Current Polarity and Full Voltage Swing Checks

The Library Compiler tool checks the following and issues a warning message or an error message if the conditions are not met:

- Full voltage swing

The Library Compiler tool issues a warning if the final voltage is not close to the final rail voltage. Calculate this margin as a percentage of the voltage swing. For example,

$$V_{\text{margin}} = 0.05 * (V_{\text{DD}} - V_{\text{SS}})$$

For the `output_current_rise` vector,


```
if ( fabs(Vfinal-VDD) > Vmargin ) {
  /*The Library Compiler tool issues warning*/
}
```

Similarly, for the `output_current_fall` vector,

```
if ( fabs(Vfinal-VSS) > Vmargin ) {
  /*The Library Compiler tool issues warning*/
}
```

The Library Compiler tool issues an error message if the final voltage for a current vector does not reach beyond the second slew threshold; that is, the maximum of the slew thresholds for the `output_current_rise` vector, and the minimum of the slew thresholds for the `output_current_fall` vector.

For example, for the `output_current_rise` vector,

```
max = MAX (slew_lower_threshold_pct_rise,
slew_upper_threshold_pct_rise, output_threshold_pct_rise)
Verr = VSS + max * (VDD-VSS) * 0.01
if ( Vfinal < Verr ) {
  /*The Library Compiler tool issues error*/
}
```

Similarly for the `output_current_fall` vector,

```
min = MIN (slew_lower_threshold_pct_fall,
slew_upper_threshold_pct_fall, output_threshold_pct_fall)
Verr = VSS + min * (VDD-VSS) * 0.01
if ( Vfinal > Verr ) {
  /*The Library Compiler tool issues error*/
}
```

In the current format, VSS is always 0, and VDD is determined as follows for the output pin containing the `output_current_rise` and `output_current_fall` vectors:

- If the `output_signal_level` attribute exists for the pin, the `power_rail` value referenced by the `output_signal_level` attribute is applied. Otherwise, the value of the voltage attribute as defined in the default operating condition of the library is applied.
- For a vector of `output_current_rise` with time or current values (T1, I1),... (Tn, In):

$$V_{final} = VSS + (0.5/C_{out}) * (I_2 + I_1) * (T_2 - T_1) + \dots + (0.5/C_{out}) * (I_n + I_{n-1}) * (T_n - T_{n-1})$$
- For a vector of `output_current_fall` with time or current values (T1, I1),... (Tn, In):

$$V_{final} = VDD + (0.5/C_{out}) * (I_2 + I_1) * (T_2 - T_1) + \dots + (0.5/C_{out}) * (I_n + I_{n-1}) * (T_n - T_{n-1})$$

- Zero current values

If the current vectors contain 0 values, Library Compiler issues an error message.

- Peak current

If either the first or last current value in a `current` vector is the peak of the current waveform, Library Compiler issues an error message.

- The Library Compiler tool checks CCS timing and compact CCS timing for the initial current and peak current and reports an error message if the current waveform value at the beginning of the waveform is more or less equal to the peak value.

- Half-unate receiver modeling

If the timing arc does not have the full receiver modeling information, the Library Compiler tool issues a warning message.

- Nonincreasing current vector values

The Library Compiler tool checks if the time index values or the currents in the values attribute of each current vector are nonincreasing.

The Library Compiler tool issues an error message for nonincreasing timing indexes. If the adjacent currents are identical, Library Compiler issues a warning message.

Note:

For scientific notation, Library Compiler accepts only the IEEE single-precision float value from the library files. The value can be represented by `s#.#####E s##`, where `s` is +/- and `#` is an integer between 0 and 9.

- Significant digits without scientific notation

The Library Compiler tool checks whether the current vector and receiver capacitance tables have at least four significant digits. If they do not, Library Compiler issues an information message. For example, "Accuracy might be affected by not using enough significant digits."

Advanced CCS Screener Checks

The following sections describe the types of checks that the `read_lib` command performs for advanced CCS libraries:

- [Compact CCS Timing Model Checks](#)

Compact CCS Timing Model Checks

The `read_lib` command checks the `compact_lut_template` group, which is used for compact CCS timing modeling, and issues a compilation error message if the following conditions are not met:

- The `index_3` value is composed of string values for compact CCS timing data from the `base_curve_type` attribute in `base_curves_group`. The `index_3` value must contain at least these six strings: `init_current`, `peak_current`, `peak_voltage`, `peak_time`, `left_id`, `right_id`. After these six parameters are specified, you are allowed to specify more parameters.
- The `base_curves_group` is required in the template and its value should be a name of predefined `base_curve` group.
- All three variables and their indexes must be defined with right values and `index_1` and `index_2` values are nonnegative float numbers.

The `read_lib` command checks the `base_curves` group and issues a compilation error if the following conditions are not met:

- `base_curve_type` must be specified, with value of `ccs_timing_half_curve`.
- Only one `base_curve` group with `ccs_timing_half_curve` is currently allowed in one library.
- Must have only one `curve_x` defined in the group. `curve_x` must be defined before `curve_y`. Values in the array must be between 0 and 1 and increase monotonically.
- The number of data values in `curve_x` and `curve_y` must be greater than or equal to three, when you specify `base_curve_type : ccs_timing_half_curve`.
- The number of data values in `curve_x` should match that in `curve_y` without taking `curve_id` into account.
- Each `curve_id` in `curve_y` should be greater than 0 and unique in the `base_curves` group.

When reading CCS timing data, the `read_lib` command issues a compilation error if the following conditions are not met:

- Only one representation of a CCS timing driver model can exist in the library, either regular CCS timing model `output_current_{rise|fall}` or compact CCS timing model `compact_ccs_{rise|fall}`.
- In the `compact_ccs_{rise|fall}` group, at the library level, the template name should be a predefined `compact_lut_template`.
- The `init_current` value should be greater than or equal to zero in `compact_ccs_rise` group and less than or equal to zero in `compact_ccs_fall` group.
- The `peak_current` value should be greater than zero in the `compact_ccs_rise` group and less than zero in the `compact_ccs_fall` group.
- The `peak_voltage` should be between VSS and VDD.
- `left_id` and `right_id` should be integers, and they should be predefined in the `base_curves` group.

The `read_lib` command issues a warning if the following conditions are not met:

- The `max_capacitance` design rule constraint of the output pin should be less than or equal to the value represented in the `compact_ccs_{rise|fall}` group.

CCS Signal Integrity Modeling Checks

The `read_lib` command checks CCS signal integrity models and issues an error message if the following conditions are not met:

- For a library with CCS noise data, each timing group can only contain the `ccsn_last_stage` group when the `ccsn_first_stage` group is present. Otherwise, the Library Compiler tool reports an error message.
- The pin direction must be consistent with the `ccsn_first_stage` or `ccsn_last_stage` group in the pin group. This involves an input pin with a `ccsn_first_stage` group, an output pin with a `ccsn_last_stage` group, and an inout pin with both a `ccsn_first_stage` group and `ccsn_last_stage` group. Otherwise, the Library Compiler tool reports an error message.

- For pin-based CCS noise models, the pin function and `driver_type` must be consistent with the subgroups of the `ccsn_first_stage` or `ccsn_last_stage` group.

For example, the `ccsn_first_stage` or `ccsn_last_stage` group on a driver pin with `driver_type` of `pull_up` must have `stage_type` with a value of `PULL_UP`. Otherwise, the Library Compiler tool reports an error message.

- The timing arc type and sense must be consistent with the sense of the `ccsn_first_stage` or `ccsn_last_stage` group on the arc. Only combinational arcs and sequential arcs including `rising_edge`, `falling_edge`, `clear`, and `preset` can have `ccsn_first_stage` and `ccsn_last_stage` groups.

The arc sense must match the combination (chaining) of the senses of the contained `ccsn_first_stage` and `ccsn_last_stage` groups. Specifically, an inverting arc must contain only a single inverting `ccsn_first_stage` group or an inverting `ccsn_first_stage` group and a noninverting `ccsn_last_stage` group, or a noninverting `ccsn_first_stage` group and an inverting `ccsn_last_stage` group. Otherwise, the Library Compiler tool reports an error message.

- If the `is_needed` attribute is set to `false` in the `ccsn_first_stage` or `ccsn_last_stage` group, the Library Compiler tool verifies that there is no other CCS noise data within the same `ccsn_first_stage` or `ccsn_last_stage` group.
- If the `is_needed` attribute is set to `true` in the `ccsn_first_stage` or `ccsn_last_stage` group, the Library Compiler tool verifies the following:

- The `stage_type` and `is_inverting` attributes are both defined.
- If the `stage_type` attribute is set to `PULL_DOWN`, the `millier_cap_fall` attribute, `output_voltage_fall` group, and `propagated_noise_high` group are required.

The Library Compiler tool does not issue warnings if the `millier_cap_rise` attribute, `output_voltage_rise` group, and `propagated_noise_low` group exist.

- If the `stage_type` attribute is set to `PULL_UP`, then the `millier_cap_rise` attribute, `output_voltage_rise` group, and `propagated_noise_low` group are required.
The Library Compiler tool does not issue warnings if the `millier_cap_fall` attribute, `output_voltage_fall` group, and `propagated_noise_high` group exist.
- If the `stage_type` attribute is set to `BOTH`, then the following attribute or groups are required:
 - `millier_cap_rise` attribute
 - `millier_cap_fall` attribute
 - `output_voltage_rise` group
 - `output_voltage_fall` group
 - `propagated_noise_rise` group
 - `propagated_noise_high` group
- If the `is_needed` attribute is set to `TRUE` in the `ccsn_first_stage` or `ccsn_last_stage` group, the Library Compiler tool verifies the following:
 - The `dc_current` group exists.
 - The values of both `index_1` and `index_2` in the `dc_current` group need to be in increasing order.
 - For the values of both `index_1` and `index_2` in the `dc_current` group, the first entry must be less than or equal to `VSS`, and the last entry must be greater than or equal to `VDD`. Otherwise, the Library Compiler tool issues a warning and not an error message.
- When either the `millier_cap_rise` or `millier_cap_fall` attribute exist, the Library Compiler tool verifies the following:
 - The value of the attribute is greater than or equal to 0.
 - The value of the attribute is *not* greater than the library maximum capacitance. Otherwise, the Library Compiler tool issues a warning and not an error message.
- For the `output_voltage_rise` or `output_voltage_fall` group, the Library Compiler tool verifies the following:
 - It has at least one vector group.
 - The template to which the vector group contains only the following three variables as `index_1`, `index_2`, and `index_3`, respectively: `input_net_transition`, `total_output_net_capacitance`, and `time`.
 - The values of both `index_1` and `index_2` of the vector group must be positive or 0.

- The values of `index_3` of the vector group must be positive or 0 and they must be in strict increasing order.
- The voltage values (`values` attribute) of the vector group must be between VSS (inclusive) and VDD (inclusive).
- For the `propagated_noise_low` or `propagated_noise_high` group, the Library Compiler tool verifies the following:
 - It has at least one vector group.
 - The template that the vector group refers to contains only the following four variables as `index_1`, `index_2`, `index_3` and `index_4` respectively: `input_noise_height`, `input_noise_width`, `total_output_net_capacitance`, and `time`.
 - The value of `index_1` (`input_noise_height`) of the vector group must be between VSS (not inclusive) and VDD (inclusive).
 - The values of `index_2` (`input_noise_width`) of the vector group must be greater than 0 (not inclusive).
 - The values of `index_3` (`total_output_net_capacitance`) of the vector group must be greater than or equal to 0.
 - The values of `index_4` of the vector group must be positive or 0 and they must be in strict increasing order.
 - The voltage values (`values` attribute) of the vector group must be between VSS (inclusive) and VDD (inclusive).
- For a timing arc, the power supply names specified by the `output_signal_level` attribute in the `ccsn_first_stage` group and the `input_signal_level` attribute in the `ccsn_last_stage` group must be identical.
- For a cell with multiple power pins:
 - The `input_signal_level` attribute must not be specified in a `ccsn_first_stage` group for an input or inout pin.
 - The `output_signal_level` attribute must not be specified in a `ccsn_last_stage` group for an output or inout pin.
- For a cell with no power pin or a single power pin, the `input_signal_level` and `output_signal_level` attributes must not be specified in the `ccsn_first_stage` and `ccsn_last_stage` groups.
- The size of the `dc_current` table in the `ccsn_first_stage` or `ccsn_first_stage` group must be at least 6x6.

The `read_lib` command issues a warning message if either of the following conditions is not satisfied:

- Each input pin or bus must have at least one `ccsn_first_stage` group defined at the pin or bus or `ccsn_first_stage` defined at one timing group inside the pin or bus.
- Each output pin or bus must have at least one `ccsn_last_stage` group defined at the pin or bus or all of its timing groups must specify the `ccsn_first_stage` group. This applies to load, dummy, and via cells.
- For a cell with multiple power pins:
 - The `output_signal_level` attribute must be specified in a `ccsn_first_stage` group for an input or inout pin.
 - The `input_signal_level` attribute must be specified in a `ccsn_last_stage` group for an output or inout pin.

Referenced CCS Noise Modeling Checks

If either the `input_ccb` or the `output_ccb` group is present in the library, the `read_lib` command considers the library to have the referenced CCS noise model and checks the following rules and generates error messages if they are not met:

- The `ccsn_first_stage` or the `ccsn_last_stage` groups must not be specified.
- The channel-connected block `input_ccb` and `output_ccb` groups must be named.
- An `input_ccb` or `output_ccb` group that is referenced in the `propagating_ccb`, `active_input_ccb`, or `active_output_ccb` attribute must be defined only in the input pin (`related_pin`) or the output pin group respectively.
- In a `pin` group, both the `input_ccb` and `output_ccb` groups must have the same pin identifier. However, the channel-connected block group identifier must be different for these groups.
- In a `timing` group, a channel-connected block group must be referenced only once using the `propagating_ccb`, `active_input_ccb`, or `active_output_ccb` attribute.
- In a `timing` group, the `output_ccb` group must be referenced only once.
- The switching sense of the `propagating_ccb` attribute must be consistent with the timing sense.
- The `is_pass_gate` attribute must not be set to `true` in the `input_ccb` group without setting the pin-level `has_pass_gate` attribute to `true`.
- Analysis must not fail during CCS noise compilation.

- The timing arc `timing_sense` attribute value must not be `positive_unate` because this is not consistent with the `is_inverting` attribute value of the associated `input_ccb` or the `output_ccb` group.
- The `input_ccb` or the `output_ccb` group must not be defined in a sequential timing arc.
- The table size of the `dc_current` table must not be less than the required 6X6.
- The `input_ccb` group must not be specified in an output `pin` group.
- If the `is_needed` attribute is set to `false` in an `input_ccb` group, the `stage_type` attribute must not be specified in this group.
- The `stage_type` attribute value in the `input_ccb` or `output_ccb` group defined in a driver `pin` group must be the same as the `driver_type` attribute value of the pin.
- The values of both the `index_1` and `index_2` attributes in the `dc_current` group must be in increasing order.
- For the values of both the `index_1` and `index_2` attributes in the `dc_current` group, the first value must be less than or equal to VSS, and the last entry must be greater than or equal to VDD.
- A specific attribute value must be within its upper and lower bounds.
- For a pin with the `has_pass_gate` attribute set to `true`, the `input_ccb` or `output_ccb` group must have the `is_pass_gate` attribute set to `true`.
- In a pin, the `when` condition of the timing arc must match the `when` condition of the `input_ccb` or the `output_ccb` group.
- The `when` condition in the `input_ccb` or `output_ccb` group must be equal or a superset of the `when` condition in the current `timing` group.
- The `input_signal_level` and `output_signal_level` attribute values respectively defined in an `input_ccb` and an `output_ccb` group must not override the PG pin voltages.
- The signal level referenced by the `input_signal_level` or the `output_signal_level` attribute of an `input_ccb` or `output_ccb` group must be defined in a PG pin. The signal level must have a positive and non-zero voltage.
- For a cell with multiple power pins
 - The `output_signal_level` attribute must be specified in an `input_ccb` group for an input or inout pin
 - The `input_signal_level` attribute must be specified in an `output_ccb` group for an output or inout pin

If the tool detects the `input_ccb` or the `output_ccb` group in the library, it checks the following rules and generates warning messages if they are not met:

- Each `timing` group must have one of the following combinations of attributes:
 - Only the `propagating_ccb` attribute
 - The `active_input_ccb` and `active_output_ccb` attributes
 - The `propagating_ccb` and `active_input_ccb` attributes
 - The `propagating_ccb`, `active_input_ccb`, and `active_output_ccb` attributes
- An input or inout `receiver_capacitance` group can only have the `active_input_ccb` attribute.
- The `related_ccb_node` attribute values of the channel-connected block groups specified in the `propagating_ccb` attribute must match.
- The `propagating_ccb`, `active_input_ccb`, or `active_output_ccb` attribute must be specified only once. If a particular attribute is specified multiple times, the last one is kept.
- For an `input_ccb` or `output_ccb` group referenced in a `timing` group, the associated `output_voltage_rise` or `output_voltage_fall` group's:
 - `index_1` value (input slew) must match one of the `index_1` values of the driver waveform.
 - `index_2` value (`total_output_net_capacitance`) must match one of the `index_2` values of the `cell_rise` or `cell_fall` data.
- An input or inout pin must include at least one nonstatic `input_ccb` group in the `pin` group or all its timing arcs must have the CCS noise data.
- A particular value of a specific attribute must be less than or equal to the `max_capacitance` attribute value.
- An input or inout pin must have at least one nonstatic `input_ccb` group.
- There are some cells that have CCS Noise data, but no CCS Noise information is defined for the specified cell.
- The CCS noise output DC voltage swing must not be significantly different from the rail voltage swing.
- The capacitances associated with the `millier_cap_rise` and `millier_cap_fall` attributes in the `input_ccb` and `output_ccb` groups must not be zero or negative.
- Both the `millier_cap_rise` and `millier_cap_fall` attribute values defined in the `input_ccb` group must be less than the pin-level `capacitance` attribute value and greater than 10 percent of the `capacitance` attribute value.

- The ratio of the `miller_cap_rise` to the `miller_cap_fall` attribute value must be less than five.

CCS Power Modeling Checks

The following sections describe the types of checks that the `read_lib` command performs for advanced CCS power models:

- [Leakage Current Syntax Checks](#)
- [Gate Leakage Modeling in Leakage Current Syntax Checks](#)
- [Intrinsic Parasitic Model Syntax Checks](#)
- [Parasitics Model Syntax Checks](#)
- [Dynamic Power Model Syntax Checks](#)
- [Power and Ground Current Syntax Checks](#)
- [Dynamic Current Syntax Checks](#)
- [Compact CCS Power Modeling Checks](#)

Leakage Current Syntax Checks

The `read_lib` command checks the following conditions in the leakage current syntax and reports error and warning messages as applicable:

- The `when` attribute must be a valid Boolean expression with the cell's structural pins. Otherwise, the Library Compiler tool issues an error message.
- Each `when` state must be mutually exclusive in a cell, which means that only one condition defined in the `when` state can be met at any given time. Otherwise, the Library Compiler tool issues an error message.
- For a cell with a single power and ground pin, either simplified format or regular format is allowed. Otherwise, the Library Compiler tool issues an error message.

The simplified format is allowed for a cell with a single power and ground pin. For the regular format, the `pg_current` group is required within a `leakage_current` group. Simplified and regular syntax formats cannot be applied to the same cell.

- Only one default state is allowed within a cell. Otherwise, a warning message is issued, and Library Compiler stores only the last default state and discards the others.
- Regular format must be chosen for a cell with multiple power or ground pins. Otherwise, the Library Compiler tool issues an error message.

The following rules apply only for the regular format:

- A power or ground pin is required, and only one specified power and ground pin is allowed for each `pg_current` group. You must provide a valid single power or ground pin of the cell. Otherwise, the Library Compiler tool issues an error message.
- Only one power or ground pin can be omitted within a `pg_current` group under a `leakage_current` group. Otherwise, the Library Compiler tool issues an error message.
- If a power or ground pin is a power pin or internal ground pin, the value of the leakage current must be 0 or a positive floating-point number. Otherwise, the Library Compiler tool issues an error message.
- If a power or ground pin is a ground pin or internal power pin, the value of the leakage current must be 0 or a negative floating-point number. Otherwise, the Library Compiler tool issues an error message.
- If there is no `gate_leakage` group in a `leakage_current` group, the `pg_current` sum among all PG pins must be zero. One PG pin can be omitted within a `pg_current` group under a `leakage_current` group. Therefore, this rule applies only when all PG pins in a `leakage_current` group are specified.

If there is a `gate_leakage` current, the sum of all `pg_current` and all `gate_leakage` currents must be zero. This rule applies only when all PG pins in a `leakage_current` group are specified and all input and inout pins are specified in `gate_leakage` groups.

The equation for both rules is

$$0 \leq \frac{|\sum x|}{\sum |x|} \leq 1.0e-6$$

where `x` is the `leakage_current` value. If these rules are not met, the Library Compiler tool issues an error message.

- If there is more than one `pg_current` group under a `leakage_current` group, the `pg_current` group must be assigned to different power or ground pins. Otherwise, a warning message is issued to alert you that there are duplicated `pg_current` groups. In this case, only the last group is recognized by Library Compiler. Otherwise, the Library Compiler tool issues a warning message.

The following rule is applied only for the simplified format:

- The value of leakage current must be 0 or a positive floating-point number. Otherwise, the Library Compiler tool issues an error message.
- If the current waveform table is a cross table, it can contain two current waveform vectors for the same index. Because the two current vectors are associated with one current

waveform, the data is redundant. If the Library Compiler tool encounters this redundant data in the library file, it issues an error message.

- For propagating events, the sum of the reference time and the maximum NLDM delay for all `related_outputs` should be a time value that lies between the minimum and the maximum times of the current waveform. Otherwise, the Library Compiler tool issues an error message.

Gate Leakage Modeling in Leakage Current Syntax Checks

The `read_lib` command checks the following conditions under a `leakage_current` group, and reports error or warning messages if applicable.

- If there is more than one `gate_leakage` group under a `leakage_current` group, the `gate_leakage` groups must be assigned to different input or inout pins. Otherwise, a warning message is issued, and only the last group is stored by the Library Compiler tool.
- An input or inout pin is required. Only one specified pin is allowed for each `gate_leakage` group, and it must be a valid single pin of the cell. Otherwise, the Library Compiler tool issues an error message.
- The `input_high_value` attribute value must be positive or zero if it is specified. Otherwise, the Library Compiler tool issues an error message.
- The `input_low_value` attribute value must be negative or zero if it is specified. Otherwise, the Library Compiler tool issues an error message.
- The `input_high_value` attribute value must be 0.0 or omitted if the state condition implies that an input pin cannot be logic 1. Otherwise, the Library Compiler tool issues an error message.
- The `input_low_value` attribute value must be 0.0 or omitted if the state condition implies that an input pin cannot be logic 0. Otherwise, the Library Compiler tool issues an error message.
- There is no check for current conservation if `gate_leakage` is present. Library Compiler ignores and does not issue an error or warning message.

Intrinsic Parasitic Model Syntax Checks

The `read_lib` command checks the following conditions in the intrinsic model syntax and reports error or warning messages if applicable:

- The `when` attribute must be a valid Boolean expression in Library Compiler. Otherwise, the Library Compiler tool issues an error message.

- Each `when` state must be mutually exclusive in a cell, which means that only one condition defined in the `when` state can be met at any given time. Otherwise, the Library Compiler tool issues an error message.
- A power or ground pin is required and only one specified power or ground pin is allowed for each `intrinsic_capacitance` group. You must provide a valid single power or ground pin of a cell. Otherwise, the Library Compiler tool issues an error message.
- A `related_output` is required and only one specified output pin is allowed for each `intrinsic_resistance` group. You must provide a valid single logic output or inout pin. Otherwise, the Library Compiler tool issues an error message.
- Intrinsic resistance and capacitance values must be positive floating-point numbers. Otherwise, the Library Compiler tool issues an error message.
- One default state is allowed within a cell. Otherwise, a warning message is issued. The tool keeps only the last default state and discards the others.
- If there is more than one `intrinsic_capacitance` group under an `intrinsic_parasitic` group, they must contain different power or ground pins. Otherwise a warning message is issued to alert that there are duplicated `intrinsic_capacitance` groups, and only the last group is recognized by Library Compiler.
- If there is more than one `intrinsic_resistance` group under an `intrinsic_parasitic` group, they must contain a different output pin. Otherwise a warning message is issued to alert that there are duplicated `intrinsic_resistance` groups, and only the last group is recognized by the tool.
- If there are two `intrinsic_parasitic` groups in a cell, neither group has a `when` condition, and the only difference between the groups is that one group only includes a `intrinsic_resistance` subgroup and the other group only includes a `intrinsic_capacitance` subgroup, the tool issues an error message stating that you must combine the groups into one. You cannot define both `intrinsic_resistance` and `intrinsic_capacitance` in the same `intrinsic_parasitic` group.
- If two `intrinsic_parasitic` groups in the same cell have the same `when` condition, the Library Compiler tool issues an error message.
- In the `intrinsic_parasitic` group, if the template name of the `lut_values` group does not match with the template name of the `lu_table_template` group at the library level, the Library Compiler tool issues an error message.
- In the `intrinsic_capacitance` and `intrinsic_resistance` groups, if the pin specified in the `reference_pg_pin` attribute is an invalid `pg_pin`, the Library Compiler tool issues an error message.

Parasitics Model Syntax Checks

The `read_lib` command checks the following items in a parasitics model.

- If there is more than one `total_capacitance` group under an `intrinsic_parasitic` group, it must be assigned to different power and ground pins. Otherwise, a warning message is issued to alert that a duplicated `total_capacitance` group exists. In this case, the Library Compiler tool only stores the last group.
- A power or ground pin is required and only one specified pin is allowed for each `total_capacitance` group, and it must be a valid single power or ground pin of the cell. Otherwise, the Library Compiler tool issues an error message.
- If `power_cell_type` is a macro cell and there is more than one `intrinsic_parasitic` group under a cell group, a warning message is issued and only the last group counts by Library Compiler.
- No state condition is allowed in `intrinsic_parasitic` group if `power_cell_type` is a macro cell. Otherwise, the Library Compiler tool issues an error message.
- No `intrinsic_capacitance` and `intrinsic_resistance` groups are allowed in `intrinsic_parasitic` group if `power_cell_type` is a macro cell. Otherwise, the Library Compiler tool issues an error message.
- No `total_capacitance` group is allowed in the `intrinsic_parasitic` group if `power_cell_type` is a standard cell. Otherwise, the Library Compiler tool, the Library Compiler tool issues an error message.

Dynamic Power Model Syntax Checks

The `read_lib` command checks the following items in a dynamic power model.

- The `power_cell_type` attribute must be a macro cell if the `min_input_switching_count` and `max_input_switching_count` attributes are specified. Otherwise, the Library Compiler tool issues an error message.
- The `min_input_switching_count` and `max_input_switching_count` attributes must always be specified in pair within a `switching_group`. Otherwise, the Library Compiler tool issues an error message.
- The `min_input_switching_count`, `max_input_switching_count`, and the `input_switching_condition` attributes should not be specified in the same `switching_group` group. Otherwise, the Library Compiler tool issues an error message.
- If the `min_input_switching_count` and `max_input_switching_count` attributes are defined in one `switching_group` group, they must be defined for the rest of the `switching_group` groups under the same `dynamic_current` group. Otherwise, the Library Compiler tool issues an error message.

- The counts within a `dynamic_current` group should cover all number bits of input specified in `related_inputs`. Otherwise, the Library Compiler tool issues an error message.
- The counts within a `dynamic_current` group cannot overlap. Otherwise, the Library Compiler tool issues an error message.
- `min_input_switching_count` must be greater than 0. Otherwise, the Library Compiler tool issues an error message.
- The `max_input_switching_count` attribute must be greater than or equal to `min_input_switching_count` and less than or equal to the total number of bits in the input pins specified in `related_inputs`. Otherwise, the Library Compiler tool issues an error message.

Power and Ground Current Syntax Checks

The `read_lib` command checks the following conditions in power or ground current template syntax and issues an error message if any of the conditions are not satisfied:

- The last variable must be `time`. The `time` variable is required.
- `input_net_transition` and `total_output_net_capacitance` are two available values for all variables except for the last variable. They can be placed in any order.
- There can be zero or one `input_net_transition` variable.
- There can be zero, one, or two `total_output_net_capacitance` variables.

Dynamic Current Syntax Checks

The `read_lib` command checks the following conditions in the dynamic current syntax and reports an error message as applicable:

- The `when` attribute must be a valid Boolean expression in the Library Compiler tool.
- The values specified in the `when`, `related_inputs`, and `related_outputs` attributes must be mutually exclusive.
- The number of entries in the `typical_capacitances` and `related_outputs` attributes must be identical.
- If `related_outputs` is specified, the list in the attribute must not be empty, and the entries in the list must be either output or inout pins without repetition.
- Bus and bundle can be specified in a `related_outputs` attribute only in the bit level.
- Bus and bundle can be supported in the `related_inputs` attribute.

- The list in a `related_inputs` attribute must not be empty, and the entries in the list must be either input or inout pins without repetition.

Two `dynamic_current` groups are considered to be overlapping if both of the following conditions are true:

- Two `dynamic_current` groups have at least one identical input pin in the list of their `related_inputs` attributes, and both groups have the same `related_outputs` list or both have no `related_outputs` defined.
- The `when` attributes specified within two `dynamic_current` groups are not mutually exclusive, or there is no `when` attribute specified within both groups.
- No two `dynamic_current` groups in the same cell can be in an overlapping condition as defined earlier.
- At least one `switching_group` must be defined for each `dynamic_current` group.
- At least one `pg_current` must be defined for each `switching_group` group.
- If cell type is macro, then `related_outputs` and `output_switching_condition` attributes cannot be applied.
- The values in `input_switching_condition` can be either `rise` or `fall`.
- The `input_switching_condition` attribute is optional if both conditions `rise` and `fall` of the input pin can be applied for the same output condition. If it is defined, only one value is allowed.
- The number of entries in an `output_switching_condition` and `related_outputs` attribute must be identical.
- The values in `output_switching_condition` can be either `rise` or `fall`.

Two `switching_group` groups are considered to be an overlapping condition if both of the following bullets are true:

- `output_switching_condition` is undefined in both `switching_group` groups; or `output_switching_condition` is defined and the values of it are identical in two `switching_group` groups
- `input_switching_condition` is undefined on both `switching_group` groups; or the values are different but one is undefined and the other has a value `rise` or `fall`
- No two `switching_group` groups under the same `dynamic_current` group can be in an overlapping condition.
- Only one name can be labeled in `pg_current` and it must be a valid power or ground pin.
- The name specified in the index related to the output pin variable attribute must match one of the values in `related_outputs`, and it must be a valid output pin.

- The only allowed values in `power_cell_type` are `stdcell` and `macro`.
- All vectors must be specified with the same template under the same `pg_current` group.
- If the `index_output` attribute is defined in one vector, it must be defined for the rest of the vectors under the same `pg_current` group.
- If the `index_output` attribute is specified in one vector, then `typical_capacitances` must be defined under the same `dynamic_current` where vector data is defined. Otherwise, the Library Compiler tool issues an error message.
- If `index_output` attribute is specified in one of vectors within a `pg_current` group, the `related_outputs`, defined under the same `dynamic_current` where the vector is defined, must contain more than one pin.
- If no `index_output` attribute is defined in all vectors under a `pg_current` group and the template that those vectors refer to contains only one `total_output_net_capacitance`, `related_outputs`, defined under the same `dynamic_current` as where vectors are defined, must contain at least one pin.
- If the `index_output` attribute is defined in one vector, the template that applied for that vector must contain exactly one `total_output_net_capacitance` variable. Otherwise, the Library Compiler tool issues an error message.
- A cross point is defined as follows: if an `index_output` is defined in a vector, and the value of `total_output_net_capacitance` index is the same as one of the values specified in the `typical_capacitances` attribute, that value maps to one of the pins in the `related_outputs` attribute, which is the same pin as specified by the `index_output` attribute.
- If `pg_current` is represented as a sparse cross table, there must be at least one cross point in one of the vectors. Cross point is defined in the previous example.
- If `pg_current` is represented as a sparse cross table and there is no `input_net_transition` value specified, there must be at least one cross point in one of the vectors. If `input_net_transition` is specified, at least one cross point is required for each `input_net_transition` that is specified.
- If two cross point vector groups have the same `input_net_transition` index value, the `time` index values and `reference_time` values must be the same for both vectors.
- If two cross point vector groups have the same index value of `input_net_transition`, the index values of `time`, `values`, and `reference_time` must be the same for both vectors. Cross point is defined earlier.
- Under the same `pg_current` group, no two vectors can have the same `index_output` and the values of all index attributes, except for the last index (`time`).
- Only one value is allowed for all index attributes, except for the last one.

- The number of entries must be the same between time, index and values attributes.
- If a template with two `total_output_net_capacitance` variables (or dense table) is applied to one of the vectors under a `dynamic_current` group, the exact two output pins must be specified in a `related_output` attribute.
- If a template with two `total_output_net_capacitance` variables is applied to all vectors under a `pg_current` group, all possible combination of capacitances between two output pins must be specified. The table must be dense.

The following two rules are applied for all vectors with a template containing an `input_net_transition` variable under a `pg_current` group. The rule pertaining to cross points is assumed to be applied before the first of the next two rules can be considered.

If two cross points vector groups have the same index value of `input_net_transition`, index values of time, values, and `reference_time` must be the same for both vectors.

- Number of entries of `total_output_net_capacitance` values (or size of `total_output_net_capacitance`) for the same output pin or `index_output` (cross type) must be identical for all vectors that have different `input_net_transition` index values. This rule applies for all vectors with a template that contains an `input_net_transition` variable under a `pg_current` group.
- The values of `total_output_net_capacitance` for the same output pin or `index_output` (cross type) must be identical for all vectors that have different `input_net_transition` index values.

The `read_lib` command checks the following conditions in the dynamic current syntax and reports a warning message as applicable:

- At most, one `related_outputs` attribute can be specified for each `dynamic_current` group. Otherwise, a warning message is issued and only the last `related_outputs` attribute is recognized by the Library Compiler tool. All the duplicated attributes are discarded.
- Only one `related_inputs` attribute is required for each `dynamic_current` group. Otherwise, a warning message is issued and only the last `related_inputs` attribute is recognized by the Library Compiler tool. All the duplicated attributes are discarded.
- Only one `input_switching_condition` attribute is allowed within a `switching_group`. Otherwise, a warning message is issued, and the Library Compiler tool takes only the last attribute and discards the others.
- Only one `output_switching_condition` attribute is allowed within a `switching_group`. Otherwise, a warning message is issued and the tool takes only the last attribute and discards the others. If `rise` or `fall` can be applied, the `input_switching_condition` is optional.

- No two `pg_current` groups under the same `switching_group` can be labeled with the same name. Otherwise, a warning message is issued and the tool takes only the last `pg_current` group and discards the other duplicated groups.
- The `pg_current` group under the `switching_group` in the `dynamic_current` group must have adequate number of grid points:
 - If the table has only one dimension, that is, only `input_net_transition` as index, there must be at least three index values.
 - If the table has multiple dimensions, there must be at least two index values for each dimension.
- When the output switches, the peak value in a CCS power vector must not be the startpoint or the endpoint of the vector values, or the peak value must not be dominant meaning that it must be less than $1\mu\text{A}$.

The `read_lib` command does not check `pg_current` groups without the output switching. The command also ignores peak current values less than $1\mu\text{A}$.

- The `pg_current` group must have adequate value points to represent the current waveform. The group must have at least three points. For current values less than $1\mu\text{A}$ that are ignored by the `read_lib` command, the `pg_current` group must have at least two points.

Compact CCS Power Modeling Checks

The following requirements must be met in the `pg_current_template` group:

- The last `variable_*` value must be `time`. The `time` variable is required.
- The `input_net_transition` and `total_output_net_capacitance` values are available for all `variable_*` attributes except the last `variable_*`. They can be placed in any order except last.

The following requirements must be met for compact CCS power modeling:

- The last `variable_*` value must be `curve_parameters`.
- The `input_net_transition` and `total_output_net_capacitance` values are available for all `variable_*` attributes except the last `variable_*`. They can be placed in any order except last.

Electromigration Checks

The `read_lib` command checks the `em_max_toggle_rate` group and reports error messages if the following conditions are not met:

- In an electromigration group, each `em_max_toggle_rate` group must have a different value of the `current_type` attribute.
 - In an input-associated electromigration group, the `em_max_toggle_rate` group lookup table must be one-dimensional with `input_transition_time` as the index variable.
 - In an inout-associated electromigration group, the `em_max_toggle_rate` group lookup table must be one-dimensional with `input_transition_time` or `total_output_net_capacitance` as the index variable.
 - In an output-associated electromigration group, the one-dimensional lookup table of the `em_max_toggle_rate` group must have `total_output_net_capacitance` as the index variable.
 - In an output-associated electromigration group, the two-dimensional lookup table of the `em_max_toggle_rate` group must have the `total_output_net_capacitance` and `input_transition_time` index variables.
-

NLDM Timing Data and NLDM Noise Data Screener

You can use the NLDM library screener to check for NLDM timing data and NLDM noise data errors in your library models.

To invoke the NLDM timing data screener:

1. Set the `enable_timing_noise_signoff` variable:

```
lc_shell> set enable_timing_noise_signoff
1
```

2. If you want to screen for nonlinear delay data, set the `check_timing_all` variable:

```
set check_timing_all 1
```

You can also screen for nonlinear delay noise data by setting the following variables:

```
set chk_noise_exis 1
set chk_noise_range 1
set chk_noise_extpol 1
set chk_noise_polar 1
set chk_noise_neg 1
set chk_IV 1
```

```
set chk_DC 1
set chk_NIC 1
set chk_NPT 1
set chk_reg1 1
set chk_reg2 1
set chk_noise_mono 1
```

See [“Nonlinear Delay Noise Variables” on page 1-63](#) for descriptions of the nonlinear delay noise variables.

3. Run the `read_lib` command:

```
lc_shell> read_lib library_name -signoff_screening
```

Cell delay and output transition time are represented in library files as lookup tables. It is assumed that cell delay is based on input transition time and output load. The vendor runs various SPICE simulations at different slew rates and different output capacitance values to accumulate delay and transition time values. These values are then placed in four lookup tables that are indexed by input transition time and output load. The four tables for each timing arc are rise cell delay, fall cell delay, rise output transition, and fall output transition. The values for cell delay and transition time are then looked up using the input transition time and output capacitance numbers.

The screener reports warnings for the following issues, all of which can cause timing inaccuracies:

- Checks for nonmonotonic delay arcs in the library
When capacitance increases, the cell delay should increase. If the cell delay in the library does not increase with increasing capacitance, this indicates that the library was not characterized properly, and timing calculation will suffer as a result. The library screener therefore warns you when your library delay arcs are not increasing with increasing capacitance.
- Screens incorrect design rules in library (potential table extrapolations)
Only a limited number of points can be simulated in SPICE for each cell. When an input transition time and output load index, a cell delay, or output transition is between points, interpolation is performed. In general, there is a slight error between the delay and slew number derived by interpolation and the actual value, but this error is +- 1 percent. When a value is indexed that falls outside of the table's range, extrapolation is performed to determine the delay and slew. In general, the error that results is much larger than interpolation, and there can be a large discrepancy between calculated delay and slew and the actual values. The screener verifies that the default design rules, maximum transition, and min/max capacitance are set correctly so the delay calculator does not use cell values outside of the characterization range.
- Checks table size recommendation

Provide at least a 7 by 7 look-up table for delay and slew. In general, the more points in the table, the more accurate the timing calculation will be.

- Checks whether table values have enough significant digits

The slew/delay values in the table must have at least 4 significant digits or 0.1picoseconds (ps) if the library time is in ns and a 0.18-micron logic library. This guideline becomes more stringent for smaller processes.

- Checks whether the starting index point in tables is 0 or very close to 0

In regions where the input transition time is small and the output load is small, delay and slew tend to increase nonlinearly with increasing input transition and output load. To avoid errors at these nonlinear regions, specify a starting index point of 0 or very close to 0.

Nonlinear Delay Noise Variables

`chk_noise_exis`

Checks for missing noise data.

`chk_noise_range`

Checks noise tables with a less than recommended range for input noise width and height.

`chk_noise_extpol`

Checks extrapolation on noise data.

`chk_noise_polar`

Checks for polarity on I-V curves.

`chk_noise_neg`

Screens negative noise tables.

`chk_IV`

Screens I-V curves.

`chk_DC`

Checks the DC noise margin.

`chk_NIC`

Screens noise immunity curves.

chk_NPT

Screens noise propagation tables.

chk_reg1

Screens low and high noise regions.

chk_reg2

Screens below_low and above_high noise regions.

chk_noise_mono

Screens nonmonotonic noise arcs.

NLDM Noise Data Screener

Use the noise data checker to screen for errors in your library noise data. To invoke the screener:

```
lc_shell> enable_timing_noise_signoff = 1
lc_shell> read_lib library_name -signoff_screening
```

The screener checks noise data to ensure it meets the requirements for static noise analysis. There are four types of noise data in a library: DC Noise Margin, IV Curves, Noise Immunity Curve and Noise Propagation Data. All the checks work on the above four data and you have the option of verifying all or some of them. By default, noise propagation data is not checked, because it is optional. The screener provides the following checks for noise.

Basic Checks

This check scans the library for essential data that is required to do static noise analysis and reports any missing information. DC Noise Margin and IV Curves must be in any noise library. The screener reports an error if they are missing. The screener warns for missing data.

Recommended Checks

These following checks are recommended, but are not essential.

- Range Checks

Verify whether the IV Curve output voltage range index specified is large enough for wide coverage. This index must cover Vdd to 2*Vdd.
- Verify whether the noise height for the minimum characterized noise width (for the noise immunity curve) is at least VDD.

There is a default tolerance of 5 percent VDD for this check.

- Verify whether the maximum characterized noise width for the Noise Immunity Curve is at least the value calculated using the following formula

$$\text{WidthMAX} = \text{SlewMAX} * \text{factor} / (\text{hstp} - \text{lstp})$$

where the factor is derived from SPICE simulations (default factor = 3). SlewMAX is the maximum transition design rule, hstp is high slew trip point, and lstp is the low slew trip point.

If max_transition rule is absent in the library, this check is not performed.

- Verify whether the maximum characterized input noise width for Noise Propagation Data table is at least the value calculated using the following formula

$$\text{WidthMAX} = \text{SlewMAX} * \text{factor} / (\text{hstp} - \text{lstp})$$

where the factor is derived from SPICE simulations (the default factor = 3). SlewMAX is the max transition design rule, hstp is high slew trip point, and lstp is low slew trip point.

If the max_transition rule is absent in the library then this check is not performed

- Verify whether the maximum characterized input noise height for noise propagation tables is at least VDD

Extrapolation Checks

All extrapolation checks contain an option for you to specify for tolerance. Only data that exceeds the specified tolerance value is reported. The extrapolation checks are as follows:

- Check if noise immunity curves do not curl upward or level off at the end. There is a tolerance of 3 percent VDD for the curling-up check and 5 percent VDD for the leveling-off check.
- Noise Propagation data table should be such that an input noise of zero height or zero width should not propagate any noise at the output. The screener extrapolates Noise Propagation Data at zero input noise height/width and reports if the extrapolated value is greater than 0.

The following tolerances are used:

```
Output noise height @ zero input noise height
<= 3%Vdd
Output noise width @ zero input noise height <= 0.1 ns

Output noise height @ zero input noise width <= 5%Vdd

Output noise width @ zero input noise width <= 0.1 ns
```

Current Polarity

The current polarity check is as follows:

- Verify if all the characterized steady state current (IV Curves) have the correct polarity. Current flow into the cell is defined as positive.

Negative Table Values

- Report any noise immunity or propagation table that has negative values in it.
- Report any negative height or area coefficients in the hyperbolic noise immunity function

Advanced Checks

Advanced Checks reports any nonmonotonic noise propagation table. For any given input noise width and height index, if the output load is increased the output noise (width and height) must decrease. There is a tolerance of 1 percent VDD.

You can also view a plot of noise height versus load for any reported noise table.

2

Generating Library Reports

You use the `report_lib` command to generate reports that contain library information. The following sections describe how to use the `report_lib` command options to generate the default report and the various specific reports.

This chapter includes the following sections:

- [Overview](#)
- [Generating Default Reports](#)
- [Generating Specific Reports](#)
- [Timing Reports](#)
- [Noise Reports](#)
- [Power Reports](#)
- [Electromigration Reports](#)
- [Functionality Reports](#)
- [Physical Library Reports](#)
- [FPGA Reports](#)
- [User-Defined Data Reports](#)

- [Job Completion Record \(JCR\) Reports](#)
- [Comprehensive Reports](#)
- [Generating Reports in HTML Format](#)

Overview

You can run the `report_lib` command without any options to generate a default report, or you can specify options to control what is included in the report. When no command options are specified, the `report_lib` command displays the library-level data and information about the available cells, including

- Header information
- Operating conditions
- Wire load models
- Design rules
- A list of cells and their types

The command options let you control the reporting of

- Timing data
- Power data
- Electromigration data
- Noise data
- Functionality data
- User-defined data
- Job completion record (JCR) data
- Comprehensive data

Generating Default Reports

If you do not specify any command options, the `report_lib` command displays a default report containing the library-level data and information about the available cells, such as the operating conditions, wire load models, design rules, types of cells, and Liberty variation format (LVF) models. [Example 2-1](#) shows the header section of a default report.

Example 2-1 Timing Report Header

```
*****
Report : library
Library: my_lib
Version: 2000.01
Date   : Tues Jan 14 14:05:20 2000
*****
```

```

Library Type           : Technology
Tool Created           : 2000.01
Date Created           : Not Specified
Library Version        : Not Specified
Time Unit              : Not specified
Capacitive Load Unit  : 1.000000pf
Pulling Resistance Unit : 1kilo-ohm
Voltage Unit           : 1V
Current Unit           : 1mA
Leakage Power Unit     : Not specified.
Bus Naming Style       : %s[%d] (default)

```

Operating Conditions:

Name	Library	Calc_mode	Process	Temp	Volt	Interconnect
NOM	my_lib	1.00	25.00	3.30	balanced_tree	
WCCOM	my_lib	1.31	70.00	3.13	balanced_tree	
WCIND	my_lib	1.31	85.00	3.13	balanced_tree	

Input Voltages:

No input_voltage groups specified.

Output Voltages:

No output_voltage groups specified.

Wire Loading Model:

```

Name           : BOX0
Location        : my_lib
Resistance      : 0
Capacitance     : 0
Area           : 0
Slope          : 0
Fanout   Length  Points Average Cap Std Deviation
-----
1           0.00
2           0.00
3           0.00
4           0.00

```

Wire Loading Model Mode: enclosed.

Porosity information:

No porosity information specified.

In_place optimization mode: match_footprint

Timing Ranges:

No timing ranges specified.

Delay Threshold Trip-Points:

input_threshold_pct_rise: 50

```

output_threshold_pct_rise:      50
input_threshold_pct_fall:      50
output_threshold_pct_fall:     50

```

```

Slew Threshold Trip-Points:
  slew_lower_threshold_pct_rise: 20
  slew_upper_threshold_pct_rise: 80
  slew_lower_threshold_pct_fall: 20
  slew_upper_threshold_pct_fall: 80

```

```
slew_derate_from_library:      1
```

Components:

Attributes:

```

af - active falling
ah - active high
al - active low
ao - always on
ar - active rising
b - black box (function unknown)
ce - clock enable
ccs - expand CCS timing modeling
cccs - compact CCS timing modeling
ccsnf - ccs noise first stage
ccsnl - ccs noise last stage
d - dont_touch
gicg - generic integrated clock-gating cell
isoe - isolation cell enable pin
lse - level shifter enable pin
mo - map_only
p - preferred
r - removable
ret - retention cell
ret(zpr) - zero-pin retention cell
s - statetable
sa0 - dont_fault stuck-at-0
sa1 - dont_fault stuck-at-1
sa01 - dont_fault both stuck-at-0 and stuck-at-1
switch_cg - switch cell (coarse grain)
sz - use_for_size_only
t - test cell
u - dont_use
udp - usable for datapath generator
va - variation-aware CCS timing modeling
val - variation-aware CCS Leakage Power modeling

```

Cell	Footprint	Attributes
Cell1	"afhcin"	d, r, udp

Operating Condition Information

When you specify the `-operating_conditions` and `-op_cond_name` options, the `report_lib` command reports the following information:

- `-operating_conditions`
Lists all of the operating conditions modeled in the logic library.
- `-op_cond_name op_cond_name`
Displays an operating condition specified in the logic library.

The `-op_cond_name` option must be followed by a specific operating condition name in the library. If you only specify the `-operating_conditions` and `-op_cond_name` options, `report_lib` reports operating condition information but does not display other information, such as template, unit, library header information, and so on.

To display other data in addition to library operating condition information, specify an additional option, such as `-timing` or `-all` with the `-operating_condition` and `-op_cond_name` options.

Generating Specific Reports

This section describes how to use the `report_lib` command options to generate specific reports. It includes the following sections:

- [Timing Reports](#)
- [Noise Reports](#)
- [Power Reports](#)
- [Electromigration Reports](#)
- [Functionality Reports](#)
- [Physical Library Reports](#)
- [FPGA Reports](#)
- [User-Defined Data Reports](#)
- [Job Completion Record \(JCR\) Reports](#)
- [Comprehensive Reports](#)

Timing Reports

You can use the following options to generate timing reports for the generic CMOS delay model, or the CMOS nonlinear delay model:

`-timing`

Displays pin-level timing information in terms of constraints or delays.

Note:

The `-timing` option becomes available only after you have read the library source file (.lib) into the tool's memory.

`-timing_arcs`

Displays all the timing arcs information in the library. When you use this option, the report lists the arc sense, type, from and to pins, and the `when` statement of all cells.

`-timing_label`

Displays all the timing arc label information.

Library Defaults

The Synopsys design tool applies default values to timing-related attributes that are not defined in the logic library. The library defaults section of a default report lists timing-related attributes and the design rule constraint default values.

For timing-related attributes defined in the library, the default report shows the corresponding default values. For timing-related attributes that are not defined in the library, the default report shows a blank value field.

[Example 2-2](#) shows the Timing Attribute Defaults section of a default report.

Example 2-2 Timing Attribute Defaults

Timing Attribute Defaults:

Attribute	Default

max_transition	
max_fanout	
fanout_load	1
max_capacitance	3.5
rise_pin_resistance	0
fall_pin_resistance	0
rise_delay_intercept	0
fall_delay_intercept	0
wire_capacitance	0

rise_wire_resistance	0
fall_wire_resistance	0
rise_nonpaired_twin	0
fall_nonpaired_twin	0

Library-Level Templates

The nonlinear delay and load-dependent delay models have library-level templates, which are written out in the timing report.

Lookup Table Template

This is the format of the lookup table template information:

```
Template_name
-----
name of the template
VARIABLE_1: variable_1
VARIABLE_2: variable_2
VARIABLE_3: variable_3
INDEX_1: float ... float
INDEX_2: float ... float
INDEX_3: float ... float
```

variable_2 and *variable_3* are optional for all tables except the three-dimensional tables used to model output load. The lines containing INDEX_2 and INDEX_3 data are optional.

Template Examples

[Example 2-3](#), [Example 2-4](#), and [Example 2-5](#) show the templates for the nonlinear delay and load-dependent delay models, respectively.

Example 2-3 Nonlinear Delay Model Template

```
Template_name
-----
prop
VARIABLE_1: total_output_net_capacitance
VARIABLE_2: input_net_transition
INDEX_1: 0.0000 1.0000 2.0000 3.0000 4.0000
INDEX_2: 0.0000 1.0000 2.0000 3.0000 4.0000
tran
VARIABLE_1: total_output_net_capacitance
VARIABLE_2: input_net_transition
INDEX_1: 0.0000 1.0000 2.0000 3.0000 4.0000
```

Example 2-4 Load-Dependent Template

```
Template_name
```

```

-----
load
VARIABLE_1: constrained_pin_transition
VARIABLE_2: related_pin_transition
VARIABLE_3: related_out_total_output_net_capacitance
INDEX_1: 0.1000    1.2000    2.3000    3.4000
INDEX_2: 0.1000    1.2000    2.3000    3.4000
INDEX_3: 0.1000    1.2000    2.3000    3.4000

```

Cell-Level Data

The timing report contains the following area and timing information for each cell.

The timing report includes a `PIN` section for every pin in the cell.

Syntax

```

CELL(cell_name) : area, ...;
  PIN(pin_name) : attributes;
    timing_information END_PIN pin_name;
  PIN(pin_name) : attributes;
    timing_information END_PIN pin_name;
END_CELL cell_name;

```

Pin Attributes

Input, output, and bidirectional pins have different pin attributes.

These are the attributes for an input pin:

```

PIN(pin_name) : in, capacitance, fanout_load,
max_transition, rise_capacitance, fall_capacitance ;

```

These are the attributes for an output pin:

```

PIN(pin_name) : out, capacitance, max_fanout, max_transition,
max_capacitance, min_fanout, min_capacitance ;

```

These are the attributes for a bidirectional pin:

```

PIN(pin_name) : inout, capacitance, fanout_load,
  max_fanout, max_transition, max_capacitance,
  min_fanout, min_transition, min_capacitance,
  rise_capacitance, fall_capacitance ;

```

Operating Conditions

Reports for scaled timing cells also include information about the `operating_conditions` group.

```
SCALED_CELL(cell_name, op_conditions) : area,
...
END_SCALED_CELL cell_name ;
```

Pin-Level Information

The Library Compiler tool lists timing information in terms of constraints or delays and pin capacitance ranges. Each category uses a different format. You generate such a report by using the `-timing` option of the `report_lib` command.

The Library Compiler tool also lists cell degradation information under timing, because cell degradation is modeled like a `timing` group.

Constraints

The constraints include setup and hold on sequential cells, nonsequential setup and hold, recovery time, removal, skew, and no-change.

Sequential Setup and Hold Constraint

In sequential cells, the constrained pin is normally the data input pin of a latch or flip-flop. When the clock pin event defined by the transition type occurs, the pin must maintain a stable logic 1 or 0 for the rise or fall time (`intrinsic_rise` or `intrinsic_fall` in CMOS linear delay models; `rise_constraint` or `fall_constraint` in CMOS nonlinear delay models).

Nonsequential Setup and Hold Constraint

Nonsequential setup and hold constraints differ from setup and hold constraints on sequential cells, in that the related pin is not a clock. These constraints define how long the data signal on an input pin remains stable before or after the related pin changes state.

Recovery Time Constraint

The constrained pin is usually an asynchronous control pin, such as clear or preset. When the clock pin event defined by the transition type occurs, the pin must remain inactive for the `intrinsic_rise` or `intrinsic_fall` time.

Removal Timing Constraint

This constraint defines the minimum separation between a clock active edge and an asynchronous control signal inactive edge.

Skew Constraint

This constraint defines the maximum interval allowed between two clock-edge events.

No-Change Constraint

This constraint defines the minimum period a signal must remain stable in relation to a related signal.

Clock Insertion Delay Constraint

The arrival timing path defines the minimum and maximum timing constraint for a pin that is driving an internal clock tree for each input transition.

In conditional timing checks,

- The enable condition is the `when` attribute.
- The start enable condition is the `when_start` attribute.
- The end enable condition is the `when_end` attribute.

Constraint Syntax

This is the syntax for each constraint in the timing report:

```
MPW : pin_name, polarity, 'enable_condition', value;
PERIOD : pin_name, 'enable_condition', value;
SETUP : constrained_pin, clock_pin, transition_type,
        related_output_pin, 'start_enable_condition',
        'end_enable_condition', (intrinsic_rise, intrinsic_fall);

MPW : pin_name, polarity, 'enable_condition', value;
PERIOD : pin_name, 'enable_condition', value;
SETUP : constrained_pin, clock_pin, transition_type,
        related_output_pin, enable_condition, (intrinsic_rise,
        intrinsic_fall);

HOLD : clock_pin, transition_type, constrained_pin,
        related_output_pin, 'start_enable_condition',
        'end_enable_condition', (intrinsic_rise, intrinsic_fall);
NON_SEQ_SETUP : constrained_pin, clock_pin, transition_type,
        related_output_pin, 'start_enable_condition',
        'end_enable_condition', (intrinsic_rise, intrinsic_fall);

HOLD : clock_pin, transition_type, constrained_pin,
        related_output_pin, 'start_enable_condition',
        'end_enable_condition', (intrinsic_rise, intrinsic_fall);
NON_SEQ_SETUP : constrained_pin, clock_pin, transition_type,
        related_output_pin, enable_condition,
        (intrinsic_rise, intrinsic_fall);
```

```

NON_SEQ_HOLD : clock_pin, transition_type, constrained_pin,
  related_output_pin, 'start_enable_condition',
  'end_enable_condition', (intrinsic_rise, intrinsic_fall);

NON_SEQ_HOLD : clock_pin, transition_type, constrained_pin,
  related_output_pin, enable_condition, (intrinsic_rise,
intrinsic_fall);

RECOV : constrained_pin, pin_transition, clock_pin,
  clock_transition, related_output_pin,
  'start_enable_condition', 'end_enable_condition',
  (intrinsic_rise, intrinsic_fall);

RECOV : constrained_pin, pin_transition, clock_pin,
  clock_transition, related_output_pin, enable_condition,
  (intrinsic_rise, intrinsic_fall);

REMOVAL : clock_pin, clock_transition, constrained_pin,
  pin_transition, related_output_pin,
  'start_enable_condition', 'end_enable_condition',
  (intrinsic_rise, intrinsic_fall);

REMOVAL : clock_pin, clock_transition, constrained_pin,
  pin_transition, related_output_pin, enable_condition,
  (intrinsic_rise, intrinsic_fall);

SKEW : triggering_clock_pin, triggering_clock_transition,
  constrained_clock_pin, constrained_clock_transition,
  related_output_pin, 'start_enable_condition',
  'end_enable_condition', (intrinsic_rise, intrinsic_fall);

SKEW : triggering_clock_pin, triggering_clock_transition,
  constrained_clock_pin, constrained_clock_transition,
  related_output_pin, enable_condition, (intrinsic_rise,
intrinsic_fall);

NOCHANGE : constrained_pin, constrained_pin_level,
  related_pin, related_pin_level, related_output_pin,
  'start_enable_condition', 'end_enable_condition',
  (intrinsic_rise, intrinsic_fall);

NOCHANGE : constrained_pin, constrained_pin_level,
  related_pin, related_pin_level, related_output_pin,
  enable_condition, (intrinsic_rise, intrinsic_fall);

CLOCK_TREE : clock_pin, constraint_type, timing_sense,
  (intrinsic_rise, intrinsic_fall), (slope_rise, slope_fall),
  (rise_resistance, fall_resistance) ;

```

Constraint Table Syntax

In the constraint table syntax, the template name can be either `rise_constraint` or `fall_constraint`. `INDEX_1`, `INDEX_2`, and `INDEX_3` appear only when the local table is defined to overwrite the value in the original template.

```
table (template_name) :
  INDEX_1 : float ... float
  INDEX_2 : float ... float
  INDEX_3 : float ... float
  VALUES : float ... float
```

[Example 2-5](#) and [Example 2-6](#) show pin constraints in different models. [Example 2-7](#) shows a report for a cell with nonsequential setup and hold constraints.

Example 2-5 Pin Constraints in a CMOS Linear Delay Model

```
PIN(D) : in, 1, , , ;
  SETUP : D, CP, rise, '', '', (1.3, 1.3);
  HOLD : CP, rise, D, '', '', (0.3, 0.3);
END_PIN D;

PIN(CD) : in, 2, , , ;
  RECOV : CD, rise, CP, rise, '', '', (0.5, 0);
END_PIN CD;

PIN(EN) : in, 3, , , ;
  NOCHANGE : EN, low, CLK, high,, '', '', (1.0,1.0)
END_PIN EN;

PIN(CLK) : in, 0, , , ;
  MPW : CLK, 1, '', 3;
  MPW : CLK, h, '', 3;
END_PIN CLK;
```

Example 2-6 Pin Constraints in a CMOS Nonlinear Delay Model

```
PIN(D) : in, 1, , , ;
  SETUP : D, CP, rise, '', '', ( , );
  rise_constraint ( scalar ) :
    VALUES : 1.3
  fall_constraint ( scalar ) :
    VALUES : 1.3
  HOLD : CP, rise, D, '', '', ( , );
  rise_constraint ( scalar ) :
    VALUES : 0.3
  fall_constraint ( scalar ) :
    VALUES : 0.3
END_PIN D;

PIN(CD) : in, 2, , , ;
  RECOV : CD, rise, CP, rise, '', '', ( , );
```

```

        rise_constraint (scalar) :
            VALUES : 0.5
    END_PIN CD;

    PIN(EN) : in, 3, , , ;
    NOCHANGE : EN, low, CLK, high, , '', '', (,);
    rise_constraint (scalar) :
        VALUES : 2.98000
    fall_constraint (scalar) :
        VALUES : 0.9800
    END_PIN EN;

    PIN(CLK) : in, 0, , , ;
    MPW : CLK, 1, '', 3;
    MPW : CLK, h, '', 3;
    END_PIN CLK;

```

Example 2-7 Nonsequential Setup and Hold Constraints

```

    PIN(d): in, 4, , , ;
    NON_SEQ_HOLD: sel, rise, d, so, '', '', ( , );
    rise_constraint ( constraint ) :
        VALUES : 1.5000

    fall_constraint ( constraint ) :
        VALUES : 1.5000

    NON_SEQ_SETUP: d, sel, rise, so, '', '', ( , );
    rise_constraint ( constraint ) :
        VALUES : 1.5000

    fall_constraint ( constraint ) :
        VALUES : 1.5000

    END_PIN d;

```

Delay Reports

The Library Compiler tool reports delays for combinational timing arcs and for sequential timing arcs.

Linear Combinational Delay

This is the format for the delay report for standard combinational timing:

```

DELAY:from_pin, to_pin, timing_type, timing_sense,
'enable_condition', (intrinsic_rise, intrinsic_fall),
(slope_rise, slope_fall), (rise_resistance,
fall_resistance);

```


The `intrinsic_rise` and `intrinsic_fall` attributes have values if specified, but the `slope_rise`, `slope_fall`, `rise_resistance`, and `fall_resistance` attributes should be empty.

Piecewise Linear Combinational Delay

The delay report for piecewise linear combinational timing has the following format:

```
DELAY: from_pin, to_pin, timing_type,
timing_sense, 'enable_condition', (intrinsic_rise,
intrinsic_fall), (slope_rise, slope_fall),
(rise_delay_intercept, fall_delay_intercept),
(rise_pin_resistance, fall_pin_resistance),
(rise_wire_resistance, fall_wire_resistance),
(rise_nonpaired_twin, fall_nonpaired_twin);
```

If attributes are composed of pieces, each attribute type is enclosed in angle brackets and, within each attribute type, each piece is separated by a colon, as shown in [Example 2-8](#).

Example 2-8 Piecewise Linear Combinational Timing Pin Delays Report

```
PIN(X) : out, 0, 20, , , 0.83, (0.02, 0.01), (0.045, -0.015);
DELAY : A1, X, prop, neg_unate, '', (0.12, 0.1), ( , ),
(<0 : 0.003 : -0.063>, <0 : -0.055 : -0.088>),
(<0.006 : 0.006 : 0.006>, <0.011 : 0.011 : 0.011>),
(<0.34 : 0.325 : 0.435>, <0.33 : 1.105 : 1.16>),
(< : : >, < : : >);
END_PIN X;
```

Nonlinear Combinational Delay

For the nonlinear delay model, the Library Compiler tool reports the table information for combinational timing attributes.

The `intrinsic_rise` and `intrinsic_fall` attributes have values if specified, but the `slope_rise`, `slope_fall`, `rise_resistance`, and `fall_resistance` attributes should be empty.

In the delay table syntax, the template name can be either `rise_propagation`, `fall_propagation`, `cell_rise`, `cell_fall`, `rise_transition`, or `fall_transition`. You use `index_1` or `index_2` only when you define the local table to overwrite the index value in the original template.

Rise and Fall Values

Rise and fall values have different definitions that depend on the timing type:

- For a combinational prop delay, a rise is the 0-to-1 transition and a fall is the 1-to-0 transition.

- If from_pin is a three-state device-enable pin, a prop delay rise is the Z-to-1 transition, and a fall is the Z-to-0 transition.
- For a disable delay, from_pin is a three-state device-enable pin. A rise is the 0-to-Z transition of to_pin; a fall is the 1-to-Z transition.
- For a clear delay, only the fall is considered. A fall is the 1-to-0 transition of the to_pin. The rise event is ignored.
- For a preset delay, only the rise is considered. A rise is the 0-to-1 transition of the to_pin. The fall event is ignored.

[Example 2-9](#) and [Example 2-10](#) show two types of reports for combinational pin delays.

Example 2-9 Standard Combinational Pin Delays Report

```
PIN(Q) : out, , , , , ( , ), ( , );
SAMPLE : CP, rise, Q, '', (1.17, 1.35), (0, 0), (0.1523, 0.0589);
DELAY : SD, Q, preset, neg_unate, '', (0.97, 1), (0, 0), (0.1523, 0);
DELAY : CD, Q, clear, pos_unate, '', (1, 0.75), (0, 0), (0, 0.0589);
END_PIN Q;
```

Example 2-10 Combinational Pin Delays Report With a Nonlinear Delay Model

```
PIN(Q) : out, , , , , ( , ), ( , );
DELAY : SD, Q, preset, neg_unate, '', ( , ), ( , ), ( , );
    rise_propagation ( test ) :
        VALUES : 1.0, 2.1, 3.4
    rise_transition ( test ) :
        VALUES : 0.5, 0.7, 1.9
DELAY : CD, Q, clear, pos_unate, '', ( , ), ( , ), ( , );
    fall_propagation ( test ) :
        VALUES : 1.0, 2.1, 3.4
    fall_transition ( test ) :
        VALUES : 0.5, 0.7, 1.9
END_PIN Q;
```

Linear Sequential Delay

This is the format for a standard sequential delay report:

```
SAMPLE : clock_pin, transition_type, to_pin,
'enable_condition', (intrinsic_rise, intrinsic_fall),
(slope_rise, slope_fall), (rise_resistance,
fall_resistance);
```

[Example 2-11](#) shows standard sequential pin delays.

Example 2-11 Standard Sequential Pin Delays

```
PIN(Q) : out, , , , , ( , ), ( , );
SAMPLE : CP, rise, Q, '', (1.16, 1.44), (0, 0), (0.0653, 0.0347);
END_PIN Q ;
```

Piecewise Linear Sequential Delay

This is the format for a piecewise linear sequential delay report:

```
SAMPLE : clock_pin, transition_type, to_pin,
'enable_condition', (intrinsic_rise, intrinsic_fall),
(slope_rise, slope_fall), (rise_delay_intercept,
fall_delay_intercept), (rise_pin_resistance,
fall_pin_resistance), (rise_wire_resistance,
fall_wire_resistance), (rise_nonpaired_twin,
fall_nonpaired_twin);
```

Nonlinear Sequential Delay

This is the format for a nonlinear sequential delay report:

```
SAMPLE : CK, rise, Q, '', ( , ), ( , ), ( , ) ;
  rise_propagation ( Template8 ) :
    VALUES : 1.0394  1.1740  1.6733  1.0544  1.1897  1.6885
              1.0516  1.1868  1.6852  1.0360  1.1709  1.6693
              1.0024  1.1373  1.6356
  fall_propagation ( Template8 ) :
    VALUES : 1.4165  1.5473  2.0101  1.5046  1.6341  2.0969
              1.5584  1.6878  2.1506  1.6100  1.7402  2.2030
              1.6393  1.7694  2.2322
  rise_transition ( Template1 ) :
    VALUES : 0.0606  0.6828

  fall_transition ( Template1 ) :
    VALUES : 0.1596  0.6372
```

Pin Capacitance Ranges

This is the format for a pin capacitance ranges report:

```
PIN(A): in, float, , , float, float;
  RISE_CAPACITANCE_RANGE(float, float)
  FALL_CAPACITANCE_RANGE(float, float)
END_PIN A ;
```

Cell Degradation Design Rule

The cell degradation design rule defines the maximum capacitive load a cell can drive without cell performance degradation.

This is the syntax for the cell degradation information:

```
cell_degradation (template_name) ;
  INDEX_1 (float ... float) ;
  VALUES : (float ... float) ;
```

This is an example of the cell degradation information in a library report after you issue the `report_lib -timing` command:

```
cell_degradation (template_1) :
  INDEX_1:1.0      2.5      3.75
  VALUES: 1.0394  1.1740  1.6733
           1.0516  1.1868  1.6852
           1.0024  1.1373  1.6356
```

Timing Arcs Report

Use the `-timing_arcs` option to generate a report like the one shown in [Example 2-12](#).

Example 2-12 Reporting Timing Arcs Information

```
lc_shell> report_lib -timing_arcs
```

```
...
Attributes:
  af - active falling
  ah - active high
  al - active low
  ar - active rising
  b  - black box (function unknown)
  ce - clock enable
  d  - dont_touch
  mo - map_only
  p  - preferred
  r  - removable
  s  - statetable
  sa0 - dont_fault stuck-at-0
  sa1 - dont_fault stuck-at-1
  sa01 - dont_fault both stuck-at-0 and stuck-at-1
  sz - use_for_size_only
  t  - test cell
  u  - dont_use
```

Cell	Attributes	#	Sense/Type	Arc From	Arc Pins To	When
<hr/>						
EO		0	pos unate	A	Z	
		1	pos unate	B	Z	
		2	neg unate	A	Z	
EO1		3	neg unate	B	Z	
		0	pos unate	A	Z	
		1	pos unate	B	Z	
EOa		2	neg unate	A	Z	
		3	neg unate	B	Z	
		0	pos unate	A	Z	

Timing Label Reports

The Library Compiler tool reports all labels associated with the timing arcs when you use the `-timing_label` option.

This is the format for a timing label report:

```
lc_shell> report_lib -timing_label
```

Cell	Arc Pins		Arc Labels
	From	To	
cell1	X[0]	C	X0_C
	X[1]	C	X1_C
	X[2]	C	X2_C
	X[3]	C	X3_C
	A	D	A_D

Timing Report Statistics

The summary section at the end of a library timing report contains simple statistics like the following:

```
Number of library cells: 18
Number of reported cells: 16
Number of reported scaled cells: 6
```

CCS Timing Model Information

CCS modeling supports additional driver model complexity by using a time- and voltage-dependent current source with an essentially infinite drive resistance. The driver model achieves high accuracy by not modeling the transistor behavior. Instead, it maps the arbitrary transistor behavior for lumped loads to an arbitrary detailed wire-load model.

Compact CCS timing models are also provided to reduce disk size issues as library files become larger with the expanding CCS modeling infrastructure. The `report_lib` command displays CCS timing model data in the header section in the following format:

```
Library Type           : Technology, CCS/Compact CCS
```

High-Level Information

The `report_lib` command displays the `ccs` (expand CCS timing modeling) and `cccs` (compact CCS timing modeling) cell-level attributes if a cell has been modeled using expanded CCS driver or compact CCS timing syntax:

```
cell_name attribute-list, ccs/cccs
```

Detailed Information

If you specify the `-timing` option, `report_lib` displays the pin-level compact CCS driver information and the receiver modeling tables:

```

PIN(pin_name) : direction, ...
  DELAY: ...
    compact_ccs_rise/compact_ccs_fall
  (template_name):
    base_curves_group : base_curve_name;
    INDEX_1 : value list;
    INDEX_2 : value list;
    INDEX_3 : value list;
    VALUES : value list;
  ...
END_PIN pin_name;

```

The `report_lib` command can help when debugging issues early in the design cycle. This command extracts the cell delay and output slew data from the CCS driver model and displays the delay report, which is similar to the nonlinear delay model report:

```

PIN(pin_name) : direction, ...
  DELAY: ...
    cell_rise/cell_fall (template_name):
      INDEX_1 : value list;
      INDEX_2 : value list;
      VALUES : value list;
      rise_transition/fall_transition (template_name):
        INDEX_1 : value list;
        INDEX_2 : value list;
        VALUES : value list;
      ...
END_PIN pin_name;

```

Converting CCS Data to NLDM Data

Currently, the CCS-to-NLDM (nonlinear delay model) conversion table is only displayed when `report_lib` reports timing for one cell or a set of cells. You must specify cell names when you run `report_lib` because the library file containing the CCS model is usually very large, and converting all the cells can be a time-consuming process.

Interdependent Setup and Hold Model Information

Interdependent setup and hold data modeling is provided for pessimism reduction for sign-off tools during the timing-check stage.

The `report_lib` command reports the following information:

- When you specify the `-timing_arcs` option, `report_lib` reports the interdependent ID information in the following format:

Cell	Attributes	Arc #	Sense/Type	Arc Pins From	To	When	Id
<i>cell_name</i>	<i>attr,...</i>	<i>#</i>	<i>type</i>	<i>pin_name</i>	<i>pin_name</i>	<i>cond</i>	<i>interdependent_id</i>

- When you specify the `-timing` option, `report_lib` displays the interdependent ID for `setup_rising`, `setup_falling`, `hold_rising`, and `hold_falling` timing arcs:

```
PIN(pin_name) : direction, ...
  SETUP: pin_name, related_pin_name, clk_edge,
        related_output_pin_name, 'when_start', 'when_end',
        'interdependence_id',
        (intrinsic_rise, intrinsic_fall);
  HOLD: related_pin_name, clk_edge, pin_name,
        related_output_pin_name, 'when_start', 'when_end',
        'interdependence_id',
        (intrinsic_rise, intrinsic_fall);
  ...
END_PIN pin_name;
```

Example

```
PIN(F1): in, 0, , , , ;
  SETUP: F1, K, rise, , '', '', (6, 6);/* normal setup_rising arc */
  SETUP: F1, K, fall, , '', '', (6, 6);
  HOLD: K, fall, F1, , '', '', (0, 0);
  HOLD: K, rise, F1, , '', '', (0, 0);
  SETUP: F1, K, rise, , '', '', 1, (8, 8);/* setup_rising arc with
interdependent_id = 1 */
  SETUP: F1, K, fall, , '', '', 1, (8, 8);
  HOLD: K, fall, F1, , '', '', 1, (0, 0);
  HOLD: K, rise, F1, , '', '', 1, (0, 0);
END_PIN F1;
```

Sensitization Model Information

Sensitization modeling records the full conditions used during cell characterization. This can help tools that use Liberty models to accurately correlate their models with SPICE.

When you specify the `-char` option, `report_lib` reports all characterization-related information, including predriver modeling information.

- At the library level, `report_lib` displays the sensitization group:

```
sensitization : sensitization_group_name
  pin_names : pin_name_list;
  vector : vector_id, vector_string;
```

- For cell and timing arcs, `report_lib` displays the `sensitization_master` attribute:

```
CELL(cell_name)
  sensitization_master : sensi_master_name;
  pin_name_map : pin_name_list;

PIN(pin_name)
  ARC: related_pin_name, pin_name, type, sense, sdt_str;
      sensitization_master : sensi_master_name;
```

```

        pin_name_map : pin_name_list;
        wave_rise : stimuli by sensi-vectors;
        wave_fall : stimuli by sensi-vectors;
        wave_rise_sampling_index : index value;
        wave_fall_sampling_index : index value;
        wave_rise_timing_interval : timing_interval_list;
        wave_fall_timing_interval : timing_interval_list;
    ARC: related_pin_name, pin_name, type, sense, sdt_str;
        ...
END_PIN pin_name;
...
END_CELL cell_name;

```

Predriver Model Information

In cell characterization, the shape of the waveform driving the characterized circuit can have a significant impact on the final results.

When you specify the `-char` option, `report_lib` displays predriver data modeling information, such as the following:

- The driver waveform lookup table template, which is displayed when the lookup table template information is written out. The `variable_1` value is `input_net_transition` and the `variable_2` value is `normalized_voltage`.
- The library-level `normalized_driver_waveform` table:

```

Normalized Driver Waveform Table:
Template_nameDriver    Waveform Name
-----
template_name        driver_waveform_name
INDEX_1 : value-list;
INDEX_2 : value-list;
VALUES : value-list;

```

- The driver waveform attributes at the cell and pin level:

```

CELL(cell_name) :
    DRIVER_WAVEFORM_RISE : driver_waveform_name;
    DRIVER_WAVEFORM_FALL : driver_waveform_name;
    DRIVER_WAVEFORM : driver_waveform_name;
    PIN(pin_name) :
        DRIVER_WAVEFORM_RISE : driver_waveform_name;
        DRIVER_WAVEFORM_FALL : driver_waveform_name;
        DRIVER_WAVEFORM : driver_waveform_name;
        ...
    END_PIN pin_name;
END_CELL cell_name;

```

Feedthrough Timing Arc Information

Liberty syntax allows you to specify feedthrough timing arcs, which are timing arcs between input and output pin pairs that are not isolated. In stage-based gate-level timing analysis, if there are feedthrough arcs, analysis tools need to compute the timing of multiple serial-connected nets simultaneously instead of analyzing a net only with driver and receivers.

The `report_lib` command writes out the `feed_through_type` attribute in timing arcs when timing information is displayed:

```
PIN(pin_name) : direction, ...
  DELAY: ...
    FEED_THROUGH_TYPE : enum(short, wire, gate);
  ...
END_PIN pin_name;
```

Mode Pin Information

Conditional data modeling is widely used in complicated modules. The `mode` pin attribute supports conditional modeling in pin-based CCS noise and pin-based CCS timing receiver models.

When you specify the `-noise_arcs` option, `report_lib` displays the following CCS noise conditional data modeling information with the `mode` or `when` attributes:

Cell	#	Type	CCS Noise Toggling pin	source of path	When/Mode
-----	-----	-----	-----	-----	-----
<i>cell_name</i>	#	<i>type</i>	<i>pin_name</i>	<i>pin_name</i>	<i>condition</i>
/* Example */					
LL_SAN2	0	ccsnf	A		
	1	ccsnl	Z	B	
	2	ccsnf	Z	A	

The `report_lib` command displays the following conditional data information for pin-based CCS timing receiver models:

```
PIN(pin_name) : direction, ...
  RECEIVER_CAPACITANCE: ...
  MODE (mode_name, mode_value);
  WHEN : sdt_string;
    receiver_capacitance1_rise
    receiver_capacitance1_fall
    receiver_capacitance2_rise
    receiver_capacitance2_fall
  ...
END_PIN pin_name;
```

On-Chip Variation Model Information

On-chip variation (OCV) causes variation in the timing performance of each transistor in a design. In advanced OCV models, these variations are modeled using derating factors based on the path depth and path distance of cells. In parametric OCV models, variations are modeled using random variation coefficients and path distance.

The `-timing` option of the `report_lib` command reports the OCV model data in the following format:

- Library-level groups and attributes

```
...
OCV Table Template:
  Template_name
-----
  2D_ocv_template
    VARIABLE_1 : path_depth
    VARIABLE_2 : path_distance
    INDEX_1   : 1 2 3
    INDEX_2   : 1 2 3

  1D_ocv_template2
    VARIABLE_1 : path_distance
    INDEX_1    : 1 2 3
...
OCV derate tables:
OCV derate group: locv1
  ocv_derate_factors(1D_ocv_template2) :
    RF_TYPE : rise_and_fall
    DERATE_TYPE : early
    PATH_TYPE : clock_and_data
    INDEX_1 : 100 200
    VALUES : 0.98 0.913

OCV derate group: aocv1
  ocv_derate_factors(2D_ocv_template) :
    RF_TYPE : rise_and_fall
    DERATE_TYPE : early
    PATH_TYPE : clock_and_data
    INDEX_1 : 1 2 3
    INDEX_2 : 100 200
    VALUES : 0.98 0.913 0.96 0.77 0.87 0.95

OCV Attributes:
Attribute                                     VALUE
-----
ocv_arc_depth                                1.0
default_ocv_derate_group                     advanced_ocv
default_ocv_derate_distance_group             location_based_ocv;
```

- Cell-level attributes

```
CELL(CELL1): 0;
  OCV Attributes:
    Attribute                                VALUE
    -----
    ocv_arc_depth                           2.0
    ocv_derate_group                         advanced_ocv
    ocv_derate_distance_group               location_based_ocv;
```

- Arc-level attributes

```
PIN(Z): out, 0, , 15000, , , ;
  DELAY: A1, Z, prop, pos_unate, '', ( , ), ( , ), ( , );

  OCV Attributes:
    Attribute                                VALUE
    -----
    ocv_arc_depth                           3.0
    ...
```

For more information about on-chip variation modeling, see “Defining On-Chip Variation” in the “Building Environments” chapter of the *Library Compiler User Guide*.

Noise Reports

Use the following options to generate noise reports:

`-noise`

Displays noise immunity, noise propagation, and current-voltage (I-V) characteristic information, as shown in [Example 2-13 on page 2-25](#).

`-noise_arcs`

Displays noise arc information, as shown in [Example 2-14 on page 2-27](#).

Example 2-13 Noise Immunity, Noise Propagation, and I-V Characteristics in a Noise Report

```
lc_shell> report_lib -noise
```

Noise Immunity Lookup Table Template:

```
Template_name
-----
mynoise_reject
  VARIABLE_1: input_noise_width
  VARIABLE_2: total_output_net_capacitance
  INDEX_1:   0.0000  0.1000  0.3000  1.0000  2.0000
  INDEX_2:   0.0000  0.1000  0.3000  1.0000  2.0000
mynoise_reject_outside_rail
  VARIABLE_1: input_noise_width
```

```
VARIABLE_2: total_output_net_capacitance
INDEX_1:  0.0000  0.1000  2.0000
INDEX_2:  0.0000  0.1000  2.0000
```

I-V Characteristics Lookup Table Template:

```
Template_name
-----
my_current_low
  VARIABLE_1: iv_output_voltage
  INDEX_1:   -1.0000 -0.1000 0.0000 0.1000 0.8000 1.6000 2.0000
my_current_high
  VARIABLE_1: iv_output_voltage
  INDEX_1:   -1.0000 -0.1000 0.3000 0.1000 0.5000 0.8000 1.6000 1.7000 2.0000
```

Noise Propagation Lookup Table Template:

```
Template_name
-----
my_noise_propagation
  VARIABLE_1: total_output_net_capacitance
  VARIABLE_2: input_noise_width
  VARIABLE_3: input_noise_height
  INDEX_1:   0.0000  1.0000  2.0000  3.0000  4.0000  5.0000  6.0000
  INDEX_2:   0.0000  1.0000  2.0000  3.0000
INDEX_3:   0.0000  1.0000  2.0000  3.0000
```

```
CELL(INV): 1;
```

```
PIN(A): in, 1, 1, , , ;
  hyperbolic_noise_low : (0.400000 1.100000 0.100000) ;
  hyperbolic_noise_high : (0.300000 0.900000 0.100000) ;
  hyperbolic_noise__high : (0.100000 0.300000 0.010000) ;
  hyperbolic_noise_below_low : (0.100000 0.300000 0.010000) ;
END_PIN A;
```

```
PIN(Y): out, 0, 10, , , , ;
  steady_state_resistance_low : 1100.000000 ;
  steady_state_resistance_high : 1500.000000 ;
  steady_state_resistance__high : 200.000000 ;
  steady_state_resistance_below_low : 100.000000 ;
```

```
PIN(y): in, 1, , , , ;
  noise_immunity_high ( my_noise_reject ) :
    VALUES : 1.3000  0.8000  0.7000  0.6000  0.5500  1.5000
              0.9000  0.8000  0.6500  0.6000  1.5000  0.9000
              0.8000  0.6500  0.6000  1.5000  0.9000  0.8000
              0.6500  0.6000  1.5000  0.9000  0.8000  0.6500
              0.6000
```

```
  noise_immunity_low ( my_noise_reject ) :
    VALUES : 1.5000  0.9000  0.8000  0.6500  0.6000  1.5000
              0.9000  0.8000  0.6500  0.6000  1.5000  0.9000
              0.8000  0.6500  0.6000  1.5000  0.9000  0.8000
              0.6500  0.6000  1.5000  0.9000  0.8000  0.6500
              0.6000
```

```

noise_immunity_below_low ( my_noise_reject_outside_rail ) :
  VALUES : 1.0000  0.8000  0.5000  1.0000  0.8000  0.5000
            1.0000  0.8000  0.5000

noise_immunity_high ( my_noise_reject_outside_rail ) :
  VALUES : 1.0000  0.8000  0.5000  1.0000  0.8000  0.5000
            1.0000  0.8000  0.5000

steady_state_current_low ( my_current_low ) :
  VALUES : 0.1000  0.0500  0.0000 -0.1000 -0.2500 -1.0000
            -1.8000

steady_state_current_high ( my_current_high ) :
  VALUES : 2.0000  1.8000  1.7000  1.4000  1.0000  0.5000
            0.0000 -0.1000 -0.8000

END_PIN y;
END_CELL INV;

Number of library cells: 1
Number of reported cells: 1
1

```

Example 2-14 Noise Arcs in a Noise Report

```
lc_shell> report_lib -noise_arcs
...
```

Components:

Arc Types:

```

nih - noise_immunity_high
nil - noise_immunity_low
niah - noise_immunity_high
nibl - noise_immunity_below_low
hnh - hyperbolic_noise_high
hnl - hyperbolic_noise_low
hnah - hyperbolic_noise_high
hnbl - hyperbolic_noise_below_low
ssch - steady_state_current_high
sscl - steady_state_current_low
ssct - steady_state_current_tristate
pnwh - propagated_noise_width_high
pnwl - propagated_noise_width_low
pnhh - propagated_noise_height_high
pnhl - propagated_noise_height_low
pnwah - propagated_noise_width_high
pnwbl - propagated_noise_width_below_low
pnhah - propagated_noise_height_high
pnhbl - propagated_noise_height_below_low

```

Arc	Arc Pins				
Cell	#	Type	From	To	When

```

INV      0  hnhA
          1  hnl   A
          2  hnah  A
          3  hnbl  A
          4  nih   A      Y
          5  nil   A      Y
          6  niah  A      Y
          7  nibl  A      Y
          8  ssch  Y
          9  sscl  A      Y
         10  pnwl  A      Y
         11  pnhl  A      Y
         12  pnwbl A      Y
         13  pnhbl A      Y

```

```
1
```

CCS Noise Model Reports

CCS noise modeling captures essential noise properties of digital circuits using very compact library representation. CCS noise models enable fast and accurate gate-level noise analysis while maintaining library characterization.

High-Level Information

The `report_lib` command displays the `ccsn` CCS noise data modeling attribute at the cell level:

```
cell_name attribute-list, ccsn
```

Detailed Information

When you specify the `-noise_arcs` option, `report_lib` displays the CCS-noise-related arc and pin information using the `ccsnf` (CCS noise first stage) and `ccsnl` (CCS noise last stage) attributes:

Cell	#	Type	CCS Noise Toggling pin	source of path	When/Mode
<i>cell_name</i>	<i>#</i>	<i>type</i>	<i>pin_name</i>	<i>pin_name</i>	<i>condition</i>
/* Example */					
LL_SAN2	0	ccsnf	A		
	1	ccsnl	Z	B	
	2	ccsnf	Z	A	

Power Reports

Use the following options to generate power reports:

`-power`

Displays power-related information, including internal power, leakage power, and multiple power supplies.

`-power_label`

Displays all the power label information.

Library Defaults

Power tools use the default values for power-related attributes if they are not defined in the logic library.

The library defaults section of the library report lists what default is being used for each power-related attribute. The power-related attributes not defined in the library are blank in the power report.

[Example 2-15](#) represents the power attribute defaults section in a power report.

Example 2-15 Power Attribute Defaults

Power Attribute Defaults:

Attribute	Default

default_cell_leakage_power	1
default_leakage_power_density	0

Library-Level Templates

The internal power models have library-level templates, which are written out in the power report. This is the format of the internal power lookup table template information:

Template_name

name of the template

```
VARIABLE_1: variable_1
VARIABLE_2: variable_2
VARIABLE_3: variable_3
INDEX_1 : float ... float
INDEX_2 : float ... float
INDEX_3 : float ... float
```

variable_2 and *variable_3* are optional for all tables except the three-dimensional tables used to model output load. The lines containing `INDEX_2` and `INDEX_3` data are optional.

[Example 2-16](#) shows an example of a lookup table template for the internal power models.

Example 2-16 Internal Power Template

Internal Power Lookup Table Template:

```

Template_name
-----
power_load_64
  VARIABLE_1: input_transition_time
  VARIABLE_2: total_output_net_capacitance
  INDEX_1:   0.0400  0.1200  0.2000  0.4000  0.6000  0.8000
  INDEX_2:   0.0638  0.1276  0.2552  0.5104  1.0208  2.0416
power_3_d
  VARIABLE_1: total_output_net_capacitance
  VARIABLE_2: equal_or_opposite_output_net_capacitance
  VARIABLE_3: input_transition_time
  INDEX_1:   0.1000  5.0000 20.0000
  INDEX_2:   0.1000  5.0000 20.0000
  INDEX_3:   0.1000  0.5000  0.8000

```

The Library Compiler tool lists the power information in terms of various power and current values. Each category uses a different format. The report is generated by using the `-power` option of the `report_lib` command. [Example 2-17](#) shows an example power information report.

Example 2-17 Example Power Information Report for a NAND2 Cell

Power Information:

Attributes:

```

a - average power specification
dc - dynamic current
i - internal power
ip - intrinsic parasitic
l - leakage power
lc - leakage current
rf - rise and fall power specification
va - cell-based variation-aware leakage current

```

Power						
Cell	#	Attr	Toggling pin	Source of path	when	Mode
NAND2	0	l				buf_mode: output_mode
	1	i,rf	ZN	A1		buf_mode: output_mode
	2	lc				buf: output_mode
	3	va				buf_mode: output_mode
	4	dc		Cx		buf: input_mode
	5	dc		Cy		buf: input_mode
	6	ip				buf: input_mode

Library-Level Multiple Power Supply Information

This section shows the format and provides examples of multiple power supply information at the library level, including

- `power_supply` group information
- `power_rail` information about the operating conditions

This is the report format for information about the `power_supply` group:

```
Power Supply Group
  default_power_rail : name
    Power Rail      Value
-----
    name_1          float
    name_2          float
```

This is the format for information about the power rail in the `operating_conditions` group:

```
Name      :      operating condition name
  Power Rail      Value
-----
  name_1          float
  name_2          float
```

[Example 2-18](#) shows a power supply report.

Example 2-18 Power Supply Information Report

```
Power Supply Group:
  default_power_rail : name

    Power Rail      Value
-----
    VDD1            5.0
    VDD2            3.30
```

[Example 2-19](#) shows a power rail report.

Example 2-19 Power Rails for Operating Conditions Report

```
Name      :      WCCOM
  Power Rail      VALUE
-----
  VDD1            4.80
  VDD2            2.90
```

Cell-Level Information

This is the format for cell leakage information at the cell level:

LEAKAGE_POWER : *when_string*, *value*;

[Example 2-20](#) shows a cell leakage power report.

Example 2-20 Cell Leakage Power Report

```
CELL(cell4): 2, 0.02;
  LEAKAGE_POWER : 'A', 0.05;

CELL(cell5): 2, 0.02;
  LEAKAGE_POWER : 'A' & B', 0.05;
  LEAKAGE_POWER : 'A & B'', 0.05;

PIN(A): in, 0.032, , , , ;
END_PIN A;

PIN(B): in, 0.033, , , , ;
END_PIN B;

PIN(Z): out, 0, , , 0.5742, , ;
END_PIN Z;
END_CELL cell5;
```

Multiple Power Supply Information

This is the format of multiple power supply information at the cell level:

```
Name      :      cell name
RAIL CONNECTION (CONN NAME) : RailpowerName
-----
      name_1      float
      name_2      float
```

[Example 2-21](#) shows a cell-level multiple power supply report.

Example 2-21 Multiple Power Supply Information Report

```
Name      :      IBUF1
RAIL CONNECTION (PV1) : VDD1
RAIL CONNECTION (PV2) : VDD2
```

Pin-Level Information

The power report contains the following area and power information for each cell:

```
CELL(cell_name) : area, ...;
  AREA : area_value

  PIN(pin_name_1) : attributes;
  ...power_information...
  END_PIN pin_name_1;
```

```

    PIN(pin_name_n) : attributes;
    ...power_information...
    END_PIN pin_name_n;

END_CELL cell_name;

```

[Example 2-22](#) is an example of a CMOS cell description in a power report.

Example 2-22 Cell Description in a Power Report

```

CELL(cell1): 2, 0.02;
    LEAKAGE_POWER : 'A', 0.05;
    PIN(A): in, 0.029, , , , ;
        INTERNAL_POWER: , , 'C & D';
            rise_power ( power_ramp ) :
                VALUES :      0.5342  0.7481  1.0494  1.9451  2.9057  3.8904

            fall_power ( power_ramp ) :
                VALUES :      0.5342  0.7481  1.0494  1.9451  2.9057  3.8904

    END_PIN A;
    ...
END_CELL cell1;

CELL(cell2): 2, 0.02;
    LEAKAGE_POWER : 'A' & B', 0.05;
    LEAKAGE_POWER : 'A & B'', 0.05;

    PIN(A): in, 0.032, , , , ;
    END_PIN A;

    PIN(B): in, 0.033, , , , ;
    END_PIN B;

    PIN(Z): out, 0, , , 0.5742, , ;
    END_PIN Z;
END_CELL cell2;

```

Multiple Power Supply Signal Information

The power report generates signal-level information for all pins in cells with multiple power supplies.

This is the format of the multiple power supply information at the pin level:

```

PIN (input_pin_name) :
    INPUT_SIGNAL_LEVEL (input_pin_name) :
        power_rail_name
    END_PIN (input_pin_name)
PIN (output_pin_name) :
    OUTPUT_SIGNAL_LEVEL (input_pin_name) : power_rail_name
END PIN

```

```
(output_pin_name) ;
```

If the pin is an I/O pin, the power report generates signal-level information as follows:

```
PIN (inout_pin_name) :
    INPUT_SIGNAL_LEVEL (inout_pin_name) :
power_rail_name
    OUTPUT_SIGNAL_LEVEL (inout_pin_name) ;
power_rail_name
END_PIN (inout_pin_name) ;
```

Example

```
PIN (A) :
    INPUT_SIGNAL_LEVEL (A) : VDD1
END_PIN (A) ;

PIN (Z) :
    OUTPUT_SIGNAL_LEVEL (Z) : VDD2
END_PIN (Z) ;

PIN (Z1) ;
    INPUT_SIGNAL_LEVEL (Z1) : VDD1
    OUTPUT_SIGNAL_LEVEL (Z1) : VDD2
END_PIN (Z1) ;
```

Generic Integrated Clock Gating (ICG) Cell Model Information

Integrated clock gating (ICG) cells combine the combinational and sequential elements of a clock gate in a single cell. They are used to prevent timing problems and clock glitches that are caused by poor placement of discrete clock-gating components. Generic integrated clock gating (ICG) cell support provides a generic solution for this type of cell.

High-Level Information

The `report_lib` command displays the `gicg` (generic integrated clock-gating cell) attribute at the cell level:

```
cell_name attribute-list, gicg
```

Power and Ground (PG) Pin Information

The `report_lib` command displays power and ground pin information, including the following:

- Library-level nominal voltage and voltage map information
- `-pg_pin` option information

If a library has been modeled using power and ground pin syntax, the information is reported in the `report_lib` header in the following format:

```
Library Type           : Technology, PG Pin Based
```

Library-level nominal voltage and voltage map information is displayed in the following format:

```
nom_voltage : voltage_value;
Voltage Map:voltage name : voltage value
           voltage_name : voltage_value
```

Example

```
nom_voltage : 3.0;
Voltage Map: voltage name: voltage value
VDD  : 3.0
VDD2 : 3.1
GND  : 0.0
GND2 : 0.3
```

The `-pg_pin` option displays power and ground pin information, such as the voltage name to which a signal pin has been linked and `pg_type` information:

```
CELL(cell_name) : area_value;
  PG_PIN(pg_pin_name) :
    VOLTAGE : voltage_name;
    PG_TYPE : enum(primary_power, primary_ground,
backup_power, backup_ground,
internal_power, internal_ground);
  END_PIN pin_name;
  ...
END_CELL cell_name;
```

When the `-pg_pin` option is selected, `report_lib` displays the `related_power_pin` or `related_ground_pin` associated with the signal pins and the `related_pg_pin` associated with the `leakage_power` or `internal_power` table. The information is displayed in the following format:

```
CELL(cell_name) :
  LEAKAGE_POWER : ...
    WHEN      : when_value;
    VALUE     : value;
    RELATED_PG_PIN : pg_pin_name;
PIN(pin_name) : ...
  RELATED_POWER_PIN : pg_pin_name;
  RELATED_GROUND_PIN : pg_pin_name;
  ...
  INTERNAL_POWER:
    RELATED_PG_PIN : pg_pin_name;
    ... /* leakage power table */
END_PIN pin_name;
```

For bias PG pins with insulated substrate wells, the `report_lib` command reports the `is_insulated` and `tied_to` attributes, as shown:

```
PG_PIN(bias_pg_pin_name) :
...
VOLTAGE_NAME : voltage_name;
PG_TYPE : pwell | nwell;
IS_INSULATED : true;
TIED_TO : pg_pin_name;
...
END_PG_PIN VDD
```

For more information about power and ground pins, see “Power and Ground (PG) Pins” in the “Advanced Low-Power Modeling” chapter of the *Library Compiler User Guide*.

Partial PG Pin Cell Information

The `report_lib` command displays partial PG pin cell information, including `-pg_pin` option information.

If a partial PG pin cell is modeled, the information is reported in the `report_lib` report, as shown in the following example:

Cell	Attributes
CELL_1	b, d, u, 0p0g
CELL_2	b, d, u, 0p0g
CELL_3	b, d, u, 0p0g

The `-pg_pin` option displays partial PG pin information, such as the voltage name to which a signal pin has been linked and the `pg_type` information:

```
CELL(cell_name) : area_value;
PG_PIN( pg_pin_name ) :
    VOLTAGE : voltage_name;
    PG_TYPE : enum(primary_power, primary_ground,
        backup_power, backup_ground,
        internal_power, internal_ground);
END_PIN pin_name;
...
END_CELL cell_name;
```

When the `-pg_pin` option is used, the `report_lib` command displays the `related_power_pin` or `related_ground_pin` associated with the signal pins and the `related_pg_pin` information associated with the `leakage_power` or `internal_power` table. The information is displayed in the following format:

```
CELL(cell_name) :
    LEAKAGE_POWER : ...
    WHEN : when_value;
    VALUE : value;
```

```

    RELATED_PG_PIN : pg_pin_name;
PIN(pin_name) : ...
RELATED_POWER_PIN : pg_pin_name;
RELATED_GROUND_PIN : pg_pin_name;
...
INTERNAL_POWER:
    RELATED_PG_PIN : pg_pin_name;
    ... /* leakage power table */
END_PIN pin_name;

```

To report cells with partial PG pins, use the `report_lib` command as follows:

```
lc_shell> report_lib library_name
```

where *library_name* is the library name stored in memory after the library file is read.

The Library Compiler tool reports cells with partial PG pins by using the following attributes:

- 1p0g
Reports cells with at least one power pin and no ground PG pins.
- 0p1g
Reports cells with no power pins and at least one ground PG pin.
- 0p0g
Reports cells with no power pins or ground PG pins.

[Example 2-23](#) shows the generated report:

Example 2-23 Report With Partial PG Pin Information

```

cell (diode) {
    area : 0.1;
    dont_use : true ;
    dont_touch : true ;
    interface_timing : true ;
    timing_model_type : abstracted ;
    pin (da) {
        related_ground_pin : VSS;
        direction : input ;
        capacitance : 10 ;
    }
    pin (dc) {
        direction : output ;
        related_ground_pin : VSS;
        capacitance : 10 ;
    }
    pg_pin (VSS) {
        pg_type : primary_ground;
        voltage_name : VSS;
    }
}

```

```
}
```

For more information about partial PG pin cells, see “Power and Ground (PG) Pins” in the “Advanced Low-Power Modeling” chapter of the *Library Compiler User Guide*.

Feedthrough Cell Information

The `report_lib` command displays multipin feedthrough cells and pins by default, or when you specify the `-pg_pin` or `-all` option.

[Example 2-24](#) shows the typical multipin feedthrough information reported.

Example 2-24 Reported Feedthrough Information

```
Cell Attributes:
    short - cell that contains multipin-feedthrough pins
    ...
Pin Attribute:
    short - A multipin-feedthrough pin
    ...
```

Cell	Footprint	Attributes
feedthrough		b, d, u, short

```
...
CELL(feedthrough): 1, b, d, u;
SHORT PORTS : A, Y ;
PIN(A): inout, short, 0, , , , , , , ;
END_PIN A;
PIN(Y): inout, short, 0.015, , , , , , , ;
END_PIN Y;
PIN(B): in, 0.01, , , , , ;
    RELATED_POWER_PIN : VDD
    RELATED_GROUND_PIN : VSS
END_PIN B;
PIN(C): in, 0.01, , , , , ;
    RELATED_POWER_PIN : VDD
    RELATED_GROUND_PIN : VSS
END_PIN C;
PIN(Z): out, , , , , ;
    RELATED_POWER_PIN : VDD
    RELATED_GROUND_PIN : VSS
END_PIN Z;
END_CELL feedthrough;
```

For more information, see “Feedthrough Signal Pin Modeling” in the “Advanced Low-Power Modeling” chapter of the *Library Compiler User Guide*.

Unconnected Pin Information

The `report_lib` command displays the pins that are internally not connected when you specify the `-pg_pin` or `-all` option.

[Example 2-25](#) shows the typical information reported for an internally unconnected pin.

Example 2-25 Reported Information for Internally Unconnected Pin

```
Pin Attributes:
    uc - unconnected pin

    ...
    PIN(A): input, uc, 0, , , , , , , , ;
    END_PIN A;
    PIN(Y): output, uc, 0.015, , , , , , , , ;
    END_PIN Y;
    ...
```

Silicon-on-Insulator Cell Information

The silicon-on-insulator (SOI) devices are fabricated on a silicon layer that rests upon an insulating layer on the substrate.

The `report_lib` command reports SOI information in the following format:

- The library-level `is_soi` attribute

```
...
IS_SOI : TRUE ;
```

- The cell-level `is_soi` attribute

```
Cell          Attributes
-----
cell_name     ...soi
```

For more information, see “Silicon-on-Insulator (SOI) Cell Modeling” in the “Advanced Low-Power Modeling” chapter of the *Library Compiler User Guide*.

Switch Cell Information

Switch cells are used to power down a specific logic module to save power consumption. There are two types of switch cells: fine-grain and coarse-grain.

The `report_lib` command displays the `switch_cg` (coarse-grain switch cell) attribute at the library level:

```
cell_name attribute-list, switch_cg
```

When you specify the `-switch` option, `report_lib` reports switch cell information in the following format:

```
CELL(cell_name) : area_value;
  PG_PIN(pg_pin_name) :
    ...
    SWITCH_FUNCTION : expression;
    PG_FUNCTION : expression;
  END_PIN pin_name;

PIN(pin_name) : direction, ...
SWITCH_FUNCTION : expression;
POWER_DOWN_FUNCTION : expression;
...
END_PIN pin_name;
END_CELL cell_name;
```

You can also specify the `-switch` option to report cell-level `dc_current` table information used to model the steady state current:

```
CELL(cell_name) : area_value;
dc_current(template_name) :
  RELATED_SWITCH_PIN : pin_name;
  RELATED_PG_PIN : pin_name;
  RELATED_VIRTUAL_PG_PIN : pin_name;
  INDEX_1 : value list
  INDEX_2 : value list
  VALUES : value list
...
END_CELL cell_name;
```

For more information, see “Switch Cell Modeling” in the “Advanced Low-Power Modeling” chapter of the *Library Compiler User Guide*.

Retention Cell Information

Retention cells are sequential cells that hold the state of the cell when the power is disabled, and restore the state when the power is enabled. The `report_lib` command reports the retention cell information with additional details, when the appropriate options are specified as shown:

High-Level Information

The `report_lib` command displays the `ret` attribute to represent the retention cell at the library level, in the following format:

```
cell_name attribute-list, ret
```

Detailed Information

When you specify the `-all`, `-timing`, or `-power` options, the `report_lib` command reports the following information:

- The cell-level `retention_cell` attribute

```
CELL(cell_name) : area_value;  
    RETENTION_CELL : retention_cell_style_string;  
    ...  
END_CELL cell_name;
```

- The pin-level `retention_pin` attribute

```
PIN(pin_name) : direction, ...  
    RETENTION_PIN : ( restore | save | save_restore, 0 | 1 );  
    ...  
END_PIN pin_name;
```

When you specify the `-pg_pin` option, the `report_lib` command reports the following attributes for the retention cell model:

- The cell-level `clock_condition` group and group attributes

```
CLOCK_CONDITION ( )  
    CLOCK_ON: expression  
    REQUIRED_CONDITION: expression  
    HOLD_STATE: L | H | N  
    CLOCKED_ON_ALSO: expression  
    REQUIRED_CONDITION_ALSO: expression  
    HOLD_STATE_ALSO: L | H | N  
END_CLOCK_CONDITION
```

- The cell-level `preset_condition` group and group attributes

```
PRESET_CONDITION ( )  
    INPUT: expression  
    REQUIRED_CONDITION: expression  
END_PRESET_CONDITION
```

- The cell-level `clear_condition` group and group attributes

```
CLEAR_CONDITION ( )  
    INPUT: expression  
    REQUIRED_CONDITION: expression  
END_CLEAR_CONDITION
```

- The cell-level `retention_condition` group and group attributes

```
RETENTION_CONDITION ( )  
    REQUIRED_CONDITION: expression  
END_RETENTION_CONDITION
```

- The pin-level `save_action`, `restore_action`, `save_condition`, `restore_condition`, and `restore_edge_type` attributes

```
SAVE_ACTION: L | H | R | F
RESTORE_ACTION: L | H | R | F
SAVE_CONDITION: expression
RESTORE_CONDITION: expression
RESTORE_EDGE_TYPE: Trigger | Leading | Trailing
```

For more information, see “Retention Cell Modeling” in the “Advanced Low-Power Modeling” chapter of the *Library Compiler User Guide*.

Always-On Cell Information

Always-on cells remain powered on by a backup power supply even when the main power supply is switched off. They are required for special complex low-power designs.

The `report_lib` command displays the `ao` (always-on) attribute at the cell level:

```
cell_name attribute-list, ao
```

The `report_lib` command also displays the `ao` (always-on) pin-level attribute:

```
CELL(cell_name) : area_value;
  PIN(pin_name) : direction, ..., ao
  ...
END_PIN pin_name;
END_CELL cell_name;
```

For more information, see “Always-On Cell Modeling” in the “Advanced Low-Power Modeling” chapter of the *Library Compiler User Guide*.

Antenna-Diode Cell Information

Antenna-diode cells are inserted in a design to fix process antenna violations. The `antenna_diode_type` attribute specifies the type of antenna-diode at the cell or pin level.

The `report_lib` command displays the `power`, `ground`, or `power_and_ground` type of the `antenna_diode_type` attribute both at the cell and pin levels as follows:

```
adt(p)
adt(g)
adt(pg)
```

For more information, see “Antenna-Diode Cell Modeling” in the “Advanced Low-Power Modeling” chapter of the *Library Compiler User Guide*.

Level-Shifter Cells and Isolation Cells

Level-shifter and isolation cells are used to connect a netlist to different power domains to meet design constraints.

The `report_lib` command displays level-shifter and isolation cell information at the cell level:

```
CELL(cell_name) : area_value;
  LEVEL_SHIFTER_TYPE : enum(LH | HL | HL_LH);
  INPUT_VOLTAGE_RANGE(float, float)
  OUTPUT_VOLTAGE_RANGE(float, float)
END_CELL cell_name;
```

The `report_lib` command displays the `lse` (level-shifter enable pin) and `isoe` (isolation cell enable pin) pin-level attributes:

```
PG_PIN(pg_pin_name) : scmr
  ...
END_PIN pin_name;
PIN(pin_name) : direction, ..., lse, isoe
  ...
END_PIN pin_name;
```

The `report_lib` command does not categorize level-shifter cells as overdrive level-shifter cells or enable level-shifter cells, and isolation cells as enable isolation cells.

The `report_lib` command displays the following information:

- For overdrive level shifter cells, the `input_signal_level` attribute but not the `related_power_pin` or `related_ground_pin` attributes on input pins
- For enable level-shifter cells, level-shifter cells with the `level_shifter_enable_pin` attribute
- For enable isolation cells, isolation cells with the `isolation_cell_enable_pin` attribute
- The `alive_during_power_up` attribute
- For level-shifter cells not powered by the switching domains, the `is_unconnected_pg_pin` attribute setting for the `pg_pin` group with the `std_cell_main_rail` attribute

For example,

```
PG_PIN(VDD): scmr,
  VOLTAGE_NAME: VDDL
  PG_TYPE: primary_power
  IS_UNCONNECTED_PG_PIN: true
END_PG_PIN VDD;
```

For more information, see “Level-Shifter Cells in a Multivoltage Design” in the “Advanced Low-Power Modeling” chapter of the *Library Compiler User Guide*.

Partially Powered-Down Isolation Cells

In partition-based designs with multiple power supplies and voltages, the outputs from a shut-down partition to an active partition must be maintained at predictable signal levels. An isolation cell isolates the shut-down partition from the active partition. At the pin level, partial power shut-down of the isolation cells is supported by the `permit_power_down`, and `alive_during_partial_power_down` attributes.

When you specify the `-pg_pin` option, the `report_lib` command displays the following pin-level attributes of the isolation cell modeled for partial power shut-down:

- The `permit_power_down` attribute

```
PERMIT_POWER_DOWN : TRUE;
```

- The `alive_during_partial_power_down` attribute

```
ALIVE_DURING_PARTIAL_POWER_DOWN : TRUE;
```

For more information, see “Isolation Cell Modeling” in the “Advanced Low-Power Modeling” chapter of the *Library Compiler User Guide*.

Macro Cell Isolation Information

Macro cells do not contain internal logic information. To identify a macro cell that is internally isolated, the `is_isolated` attribute is used. The `isolation_enable_condition` attribute specifies the isolation condition for the internally isolated pins, buses, and bundles of the macro cell.

When you specify the `-pg_pin` option, the `report_lib` command displays the following attributes for an internally isolated macro cell:

- The `is_isolated` attribute

```
IS_ISOLATED : TRUE | FALSE;
```

- The `isolation_enable_condition` attribute

```
ISOLATION_ENABLE_CONDITION : boolean_expression;
```

For more information, see “Isolation Cell Modeling” in the “Advanced Low-Power Modeling” chapter of the *Library Compiler User Guide*.

CCS Power Model Information

Composite current source (CCS) power modeling addresses static power, dynamic power, and dynamic voltage (IR) drops. Compared to the nonlinear power models (NLPMs), the CCS power models provide unified data modeling, multivoltage support, accuracy, and better characterization runtime.

When you specify the `-power` or `-all` options, the `report_lib` command displays the information for the leakage current, intrinsic parasitic, dynamic current, and cell-based variation-aware leakage current modeling in the following example formats:

- Leakage current modeling

```
CELL(cell_name): ...;
  LEAKAGE_CURRENT :
    WHEN : when_string
    MODE : mode_name, mode_value
    VALUE : current_value/* presence when no pg_current*/
    PG_CURRENT (pg_pin_name): pg_current_value
    GATE_LEAKAGE(pin_name): low_value, high_value
```

- Intrinsic parasitic modeling

```
CELL(cell_name): ...;
  INTRINSIC_PARASITIC :
    WHEN : when_string
    MODE : mode_name, mode_value
    INTRINSIC_RESISTANCE(pg_pin_name):
related_output_pin_name, res;
    INTRINSIC_CAPACITANCE(pg_pin_name): cap;
    TOTAL_CAPACITANCE(pg_pin_name): cap;
```

- Dynamic current modeling

```
CELL(cell_name): ...;
POWER_CELL_TYPE : enum(stdcell, macro);
DYNAMIC CURRENT:
RELATED PINS: pin_name
  RELATED PINS: pin_name
  MODE: mode_name, mode_value
  Switching Group
    Min Input Switching Count: count
    Max Input Switching Count: count
    PG_CURRENT: value
    vector ( vector_name ) :
    REFERENCE TIME: time
    INDEX_1 : value_list
    INDEX_2 : value_list
    VALUES : values
```

Note:

The Library Compiler tool reports the switching group only when you use the following option of the `report_lib` command:

```
lc_shell> report_lib library_name -power cell_name
```

The CCS power is reported only when you specify the cell name in the library.

- Cell-based variation-aware leakage current modeling

```
VA LEAKAGE CURRENT:
MODE: mode_name, mode_value
VA VALUES: (value_list)
PG_CURRENT(pg_pin_name): pg_current_value
GATE_LEAKAGE(pin_name): (low_value, high_value)
```

For more information, see the “Composite Current Source Power Modeling” chapter of the *Library Compiler User Guide*.

Nonlinear Power Model Information

When you specify the `-power` or `-all` options, the `report_lib` command reports the following leakage and internal power modeling information, in the example formats shown:

- Leakage power

```
CELL(cell_name): ...;
LEAKAGE_POWER :
  POWER_LEVEL : pg_pin_name
  WHEN : when_string;
  MODE : mode_name, mode_value
  VALUE : value
```

- Internal power

```
INTERNAL_POWER : pin_name, ''
  WHEN : when_string;
  MODE : mode_name, mode_value
  power_level : pg_pin_name
  rise_power (scalar) :
    VALUES : value

  fall_power (scalar) :
    VALUES : value
.....
```

For more information, see the “Composite Current Source Power Modeling” chapter of the *Library Compiler User Guide*.

Analog Signal Attribute Information

The pin-level `is_analog` attribute identifies the analog signal pins, buses, and bundles. When you specify the `pg_pin` option, the `report_lib` command reports the `is_analog` attribute as follows:

```
IS_ANALOG : TRUE;
```

For more information, see “`is_analog` Attribute” in the “Defining Core Cells” chapter of the *Library Compiler User Guide*.

Pad Cell Attribute Information

The `is_pad` attribute identifies a pad pin of an I/O cell.

When you specify the `-pg_pin` option, the `report_lib` command reports the `is_pad` attribute for the cell as follows:

```
IS_PAD : TRUE;
```

For more information, see “`is_pad` Simple Attribute” in the “Advanced Low-Power Modeling” chapter of the *Library Compiler User Guide*.

Electromigration Reports

Use the following option to generate an electromigration report.

```
-em
```

Displays electromigration-related information.

Library Defaults

[Example 2-26](#) shows the timing attribute defaults section in an electromigration report.

Example 2-26 Timing Attribute Defaults

```
Attribute                                Default
-----
em_temp_degradation_factor : 0.0
```

Library-Level Templates

The electromigration data has library-level templates, which are written out in the electromigration report.

This is the format of the lookup table electromigration template information:

```
Template_name
-----
name of the template
  VARIABLE_1: variable_1
  VARIABLE_2: variable_2
  INDEX_1 : float ... float
  INDEX_2 : float ... float
```

Pin-Level Information

The electromigration report includes a `PIN` section for every pin in the cell, as shown:

```
CELL(cell_name) : area, ...;
  PIN(pin_name) : attributes;
    electromigration_information
  END_PIN pin_name;
END_CELL cell_name;
```

[Example 2-27](#) shows examples of the lookup table templates for the electromigration models.

Example 2-27 Template Examples

```
CELL(example): 2;
PIN(y): out, 0, , , 1.812, , ;
ELECTROMIGRATION: a;
  em_max_toggle_rate ( output_by_cap_and_trans ) :
  VALUES :      2.0000  1.0000  0.5000  1.5000  0.7500  0.3300  1.0000  0.5000  0.1500

ELECTROMIGRATION: b;
  em_max_toggle_rate ( output_by_cap_and_trans ) :
  VALUES :      2.0000  1.0000  0.5000  1.5000  0.7500  0.3300  1.0000  0.5000  0.1500

ELECTROMIGRATION: c;
  em_max_toggle_rate ( output_by_cap_and_trans ) :
  VALUES :      2.0000  1.0000  0.5000  1.5000  0.7500  0.3300  1.0000  0.5000  0.1500

END_PIN y;
```

Functionality Reports

Functionality reports provide statetable, multibit, and VHDL name information.

Statetable Report

Use the following options to generate statetable reports.

`-table`

Displays the statetable data in a compact form. [Example 2-28](#) provides an example of this option.

`-full_table`

Displays the statetable model in a table form and memory information about the ports. [Example 2-29](#) provides an example of this option.

Note:

The statetable options are enabled only when the logic library is read in. The asterisk suffix (*) in a statetable report denotes a delayed port.

Example 2-28 Statetable Report in Compact Form

```
lc_shell> report_lib -table asic.db dffz
...
```

```
Statetable descriptions
CELL(dffz):

  PIN(QN) : CP*, QN*, D, CP
  Table : NNNNNNNNNHNLNHN
  END_PIN QN;

  PIN(Q) :
    state_function : QN'
  END_PIN Q;

  PIN(QNZ) :
    state_function : QN
    three_state : EN
  END_PIN QZ;
END_CELL dffz;
```

Example 2-29 Statetable Report in Table Form

```
lc_shell> report_lib -full_table asic.db dff
...
```

```

Statetable descriptions:
CELL(dff):
  PIN(QN) : CP*, QN*, D, CP
  Table :
    0000      N
    0001      H
    0010      N
    0011      L
    0100      N
    0101      H
    0110      N
    0111      H
    1000      N
    1001      N
    1010      N
    1011      N
    1100      N
    1101      N
    1110      N
    1111      N
  END_PIN QN;
END_CELL dff;

```

VHDL Name Report

Use the `-vhdl_name` option to generate the VHDL name report.

`-vhdl_name`

Displays the list of changed VHDL names at the end of the report. [Example 2-30](#) provides an example of this option.

Example 2-30 Table of Changed VHDL Names

```

lc_shell> report_lib -vhdl_name my_lib
...

```

Cell	Port	VHDL name
INV		
	IN	INb
XOR		XORb
TINV		T_INV
	IN	A
NAND4		NAND4b
		Ab
	B	A
	C	Cb

Test Cell Information

Additional error messages and warnings result when the timing and test cell of the sequential model are closely checked.

It is important to update the logic libraries by accurately defining the full function of each library cell and the nontest mode of each test cell. Use the statetable format and, if possible, the ff or latch format. If the ff or latch format is adequate and you do not supply statetable information, you risk specification error.

[Example 2-31](#) is an example of a test cell report.

Example 2-31 Test Cell Report

```
lc_shell> report_lib -table asic.db dff_test
...

CELL(dff_test):

  PIN(QN) : CP*, S, D0, D1, CP
  TABLE : NHNLNHNHNNNNNNNNNN
  END_PIN QN;

  TEST_CELL
  PIN(QN) : CP*, D0, CP
  TABLE : NHNLNHNHNNNNNNNNNN
  END_PIN QN;
  END_TEST_CELL;
END_CELL dff_test;
```

Multibit Report

Use the `-multibit` option to generate multibit cell information.

When you specify the `-multibit` option with the `library_name` argument, the `report_lib` command generates the following information about each multibit cell of the specified library.

- The multibit width.
- Whether the cell structure is parallel or serial.
- Whether the cell has a dedicated scan output or not.

[Example 2-32](#) is an example of a multibit information report.

Example 2-32 Multibit Information Report

```
Multi-bit cell info:
-----
```

```

CELL(SDFF2_SO):
    Multi-bit Width           : 2
    Parallel/Serial structured : Serial
    Has dedicated scan output  : True
END_CELL SDFF2_SO;

CELL(SDFF3_SO):
    Multi-bit Width           : 3
    Parallel/Serial structured : Serial
    Has dedicated scan output  : True
END_CELL SDFF3_SO;

```

Programmable Driver Type Model Information

To support pull-up and pull-down circuit structures, Liberty models for I/O pad cells support pull-up and pull-down driver information. Liberty syntax also supports conditional (programmable) pull-up and pull-down driver information.

The `report_lib` command reports programmable pull-up and pull-down information in the following format:

```

PIN(pin_name) :
    PULL_UP_FUNCTION : expression;
    PULL_DOWN_FUNCTION : expression;
    BUS_HOLD_FUNCTION : expression;
    OPEN_DRAIN_FUNCTION : expression;
    OPEN_SOURCE_FUNCTION : expression;
    RESISTIVE_FUNCTION : expression;
    RESISTIVE_0_FUNCTION : expression;
    RESISTIVE_1_FUNCTION : expression;
    ...
END_PIN cell_name;

```

Physical Library Reports

To generate reports of physical libraries, use the `-physical` option with the `report_lib` command. When you specify the `-physical` option with the library name, the tool considers the library to be a physical library and ignores all other specified options.

```

report_lib -physical {parasitic_tech | antenna | statistic | all}
               library_name

```

The `-physical all` option is identical to the `-physical {parasitic_tech, antenna, statistic}` option and enables all the three arguments.

The report header includes the library name, version and date, full library name, file name and cell count, as shown in the following example:

```
*****
Report : library
Library: lib1
Version: O-2018.06
Date   : Tue May 29 01:09:34 2018
*****
Full name: path/lib1.ndm:lib1
File name: path/lib1.ndm
Cell count: 1
```

Using the `parasitic_tech` argument reports the parasitic technology information in the specified physical library, as shown in the following example:

```
Parasitic tech data report
-----
Parasitic tech name:      tlu_lib1
Parasitic itf technology name: my_technology
Parasitic tech type:      TLUPLUS
Parasitic source file name: my_tlu.tlu
Parasitic canonical file name: path/my_tlu.tlu
```

Using the `antenna` argument reports existing antenna properties of the library cells, including cells and pins with gate area or diffusion area and diode protection antenna properties, as shown in the following example:

```
-----
** Antenna Data:
Total number of cells with gate area and diode protection antenna
properties: 6 (out of 16)
-----
```

Cell name	Cell type	Antenna property	Pin name(s)
cell1	lib_cell	gate_area	A EN (Total 2)
mypadcell	pad	diff_area	A IEN OEN Y1 (Total 4)
mypadcell	pad	model-6*	A IEN OEN Y1...(Total 150)

Using the `statistic` argument reports the physical library statistic information about sites, layers, the path, source files, cell details and pin details, as shown in the following example:

```
-----
** Physical Data:
Site Name      Width      Height      XY      Symmetry      Default
-----
```

Site Name	Width	Height	XY	Symmetry	Default
gaunit	0.8u	1.8u	X Y		false
unit	0.2u	1.8u	--		true

Layer Name	Mask Name	Direction	Pitch	Track Offset
M1	metal1	Horizontal	0.2u	--
M2	metal2	Vertical	0.2u	--
M3	metal3	Horizontal	0.2u	--
M4	metal4	Vertical	0.2u	--
M5	metal5	Horizontal	0.2u	--
M6	metal6	Vertical	0.2u	--
M7	metal7	Horizontal	0.2u	--
M8	metal8	Vertical	0.8u	--
M9	metal9	Horizontal	0.8u	--

Source File Name

FRAME *path/lib1.ndm*
 TF *path/my_tech.tif*

Summary of Library Views

timing(t): 0
 layout(l): 0
 design(d): 1
 frame(f): 1

Summary of Library Design Types

lib_cell 1

Cell Views	Cell Type	Allowable Orientation	Mask Shiftable	Site Height	Multiple	Boundary Bbox	Cell Area (um^2)
AN2	d f lib_cell	R0	true	unit 1.8u	1X	{0.0000 0.0000} {1.6000 1.8000}	2.8800

Summary of Library Pins

ground 1
 power 1
 signal 3

Cell Name: AN2.frame

Pin Name	Direction	Port Type	PG Type	Bias Type	Secondary PG/Diode
VSS	in	ground	--	--	--
Z	out	signal	--	--	--
VDD	in	power	--	--	--
A	in	signal	--	--	--
B	in	signal	--	--	--

FPGA Reports

Use the following option to generate the FPGA report:

`-fpga`

Displays the library defaults and, if they exist, the part group and the I/O attributes.

[Example 2-33](#), [Example 2-34](#), and [Example 2-35](#) show these three sections in the order they appear in a report:

Library Defaults

Example 2-33 Library Defaults Section of the FPGA Report

```
lc_shell> report_lib -fpga example
...
*****
Report : library
Version: 2003.12
Library: example
Date   : Tue Oct  2 15:22:55 2003
*****
Library Type      : FPGA Technology
Tool Created     : 2002.05
Date Created      : Not Specified
Library Version   : Not Specified
Time Unit         : lns
default_part ("AUTO", "-5")
FPGA Technology   : "my_technology"
```

Parts Information

[Example 2-34](#) shows the PART section of the FPGA report, which lists the number of pins, rows, columns, flip-flops, block RAMs, and constraints, as well as the speed grades, if they are present in the library.

Example 2-34 PART Section of the FPGA Report

```
PART(V100CS144)
  pin_count : 94;
  num_rows  : 20;
  num_cols  : 30;
  num_ffs   : 2760;
  num_blockrams : 10;
  valid_speed_grades ("-6", "-5", "-4");
  valid_step_levels ( "STEP0", "STEP1", "STEP2" )
  default_step_level : "STEP0" ;
  max_count(BUFGTS, 4);
END PART V100CS144;
```

Cell-Level Information

[Example 2-35](#) shows the information contained in the cell-level section of the FPGA report.

Example 2-35 Cell-Level Section of the FPGA Report

FPGA IO Attributes:

Cell	io_type	drive_type	slew_type

IBUF	LVTTL		
IBUF_LVCMOS2	LVCMOS2		
IBUF_PCI33_3	PCI33_3		
IBUF_PCI33_5	PCI33_5		
IBUF_PCI66_3	PCI66_3		
OBUF	LVTTL	12	fast
OBUF_S_2	LVTTL	2	fast
OBUF_S_4	LVTTL	4	none
OBUF_S_6	LVTTL	6	none

FPGA Resource Usage:

Cell	resource_name	number_resource

IBUF	DCM	6
IBUF_LVCMOS2	DCM	16
IBUF_PCI33_3	DCM	26
IBUF_PCI33_5	DCM	36
IBUF_PCI66_3	DCM	46
OBUF	DCL	6
OBUF_S_2	DCL	16
OBUF_S_4	DCL	26
OBUF_S_6	DCL	26

User-Defined Data Reports

Use the following option to generate a user-defined data report.

`-user_defined_data`

Displays all the user-defined attributes and groups in the logic library.

When you specify both the `-user_defined_data` option and a list of cells (see [“Comprehensive Reports” on page 2-60](#)), the report displays only the user-defined attributes and groups for the specified cells. [Example 2-36](#) shows a user-defined data report.

Example 2-36 User-Defined Data in a Report

```
lc_shell> report_lib -user_defined_data example
```

```
...
```

```
*****
```

```
Report : library
```

```
Library: example
```

```
Version: 2003.12
```

```
Date   : Fri Sep 26 09:11:17 2003
```

```
*****
```

```
Library Type      : Technology
```

```
Tool Created     : 2003.12
```

```
Date Created     : 06-JUL-90
```

```
Library Version  : 0.930000
```

```
User-defined Data Template:
```

```
-----
```

```
timing
```

```
  timing_attr : string
```

```
  timing_udg
```

```
    timing_udg_attr : string
```

```
pin
```

```
  float_attr : float
```

```
  pin_attr : string
```

```
  pin_udg1
```

```
    pin_udg_attr : string
```

```
  pin_udg2
```

```
    pin_udg_attr : string
```

```
cell
```

```
  cell_attr : string
```

```
  cell_udg1
```

```
    cell_udg_attr : string
```

```
  cell_udg2
```

```
    cell_udg_attr : string
```

```
library
```

```
  lib_attr : string
```

```
  float_attr : float
```

```
  lib_udg1
```

```
    lib_udg_attr : string
```

```
  lib_udg2
```

```
    lib_udg_attr : string
```

```
User-defined Data:
```

```
-----
```

```
LIBRARY(example) :
```

```
  float_attr : 1e-09
```

```
  lib_attr : lib_test
```

```
  LIB_UDG1(lib_udg1) :
```

```
    lib_udg_attr : test
```

```

END_LIB_UDG1 lib_udg1;
LIB_UDG2(lib_udg2) :
    lib_udg_attr : test
END_LIB_UDG2 lib_udg2;
CELL(INV) :
    cell_attr : inv_test
    CELL_UDG1(cell_udg1) :
        cell_udg_attr : test
    END_CELL_UDG1 cell_udg1;
    CELL_UDG2(cell_udg2) :
        cell_udg_attr : test
    END_CELL_UDG2 cell_udg2;
    PIN(A) :
        float_attr : 0.123457
        pin_attr : pin_test
        PIN_UDG1(pin_udg1) :
            pin_udg_attr : test
        END_PIN_UDG1 pin_udg1;
    END_PIN A;
    PIN(Y) :
        PIN_UDG2(pin_udg2) :
            pin_udg_attr : test
        END_PIN_UDG2 pin_udg2;
        TIMING() :
            timing_attr : timing_test
            TIMING_UDG(timing_test) :
                timing_udg_attr : timing_udg_test
            END_TIMING_UDG timing_test;
        END_TIMING ;
    END_CELL INV;
CELL(INV1) :
    CELL_UDG1(cell_udg1) :
        cell_udg_attr : test
    END_CELL_UDG1 cell_udg1;
    CELL_UDG2(cell_udg2) :
        cell_udg_attr : test
    END_CELL_UDG2 cell_udg2;
    PIN(A) :
        PIN_UDG1(pin_udg1) :
            pin_udg_attr : test
        END_PIN_UDG1 pin_udg1;
    END_PIN A;
    PIN(Y) :
        PIN_UDG2(pin_udg2) :
            pin_udg_attr : test
        END_PIN_UDG2 pin_udg2;
    END_PIN Y;
END_CELL INV1;
END_LIBRARY example;

```

Job Completion Record (JCR) Reports

Use the following option to generate a job completion record (JCR) report.

`-jcr`

Lists the job completion record information stored in the library. This option applies only to logic libraries.

When you specify the `report_lib -jcr` command, the Library Compiler tool reports its own job completion record, and the job completion record of the characterization tool if imported.

[Example 2-37](#) shows such a report example.

Example 2-37 Job Completion Record Information

```
*****
Report : library
Library: z0
Version: I-2013.12-SP
Date   : Sun Nov 17 23:08:39 2013
*****

Library Type           : Technology
Tool Created          : I-2013.12-SP
Date Created          : Not Specified
Library Version       : Not Specified

JCR Information :
-----
#SNPS_JCR_INFORMATION_BEGIN

#VERSION=1.1 JCR_CHKSUM=47660
#PRODUCT_NAME=MyProduct
#PRODUCT_VERSION=2013.06
#PRODUCT_EXEC=my_exe
#HOST=rcrhl
#DESIGN=file1.db
#NUMBER_OF_ERRORS=0
#NUMBER_OF_WARNINGS=0
#NUMBER_OF_INFO_MSG=0
#RUN_STATUS=0
#START_TIME=Mon Nov 04 22:00:21 2013
#END_TIME=Mon Nov 04 22:00:21 2013
#OUTPUT_FOLDER=
#NUMBER_OF_FILES=0
#LOG_FILE_PATH=
#WARNING_AS_ERROR=0
#ERROR_AS_WARNING=0
#NUMBER_OF_THREADS=0
#NUMBER_OF_CORES=0
#MEMORY_UTILIZATION=0
#DOWNSTREAM_PRODUCT_INFO=(name=star_rcxt), (error=10), (warning=20), (info_msg=0)
#DOWNSTREAM_PRODUCT_INFO=(name=hspice), (error=30), (warning=50), (info_msg=10)
#NOTES=

#SNPS_JCR_INFORMATION_END
```

```

-----
#SNPS_JCR_INFORMATION_BEGIN

#VERSION=1.1 JCR_CHKSUM=41970
#PRODUCT_NAME=dc_shell
#PRODUCT_VERSION=I-2013.12-SP
#PRODUCT_EXEC=/abcd/efgh/command_shell
#HOST=ijkl
#DESIGN=z0.lib
#NUMBER_OF_ERRORS=0
#NUMBER_OF_WARNINGS=9
#NUMBER_OF_INFO_MSG=0
#RUN_STATUS=0
#START_TIME=Sun Nov 17 23:08:36 2013
#END_TIME=Sun Nov 17 23:08:39 2013
#OUTPUT_FOLDER=
#NUMBER_OF_FILES=0
#LOG_FILE_PATH=
#WARNING_AS_ERROR=0
#ERROR_AS_WARNING=0
#NUMBER_OF_THREADS=0
#NUMBER_OF_CORES=0
#MEMORY_UTILIZATION=0
#NOTES=

#SNPS_JCR_INFORMATION_END

```

Comprehensive Reports

Use the following options to generate comprehensive library reports.

`-all`

Enables all report options.

`{list_of_cells}`

A list of cells separated by commas. The report includes data only for the specified set of cells.

[Example 2-38](#) shows a report that includes only the specified cells.

Example 2-38 Command for Generating a Report of a List of Cells

```
lc_shell> report_lib mylib {AND, OR}
```

```

*****
Cell      Footprint      Attributes
-----
AND
OR
*****

```

Cell Type Information

When you specify the `-all` option, the `report_lib` command displays the modeled cell types. [Example 2-39](#) shows a report that includes multibit scan cell types.

Example 2-39 Report Displaying the Scan Cell-Types

Cell	Footprint Attributes
mb16_ffqqnrnsnx1	"mb16_ffqqnrnsn" b, s
mb16_sffqqnrnsnx1	"mb16_sffqqnrnsn" r, s, t(mux_ff)

For more information, see “Describing a Multibit Scan Cell” in the “Defining Test Cells” chapter of the *Library Compiler User Guide*.

Generating Reports in HTML Format

The Library Compiler tool provides library reports in HTML format so you can link error messages and warning messages to their man page and log file for more information. To enable reports in HTML format, use the `read_lib` command with the `-html` option. The HTML reports classify messages by ID and display error messages and user-specified IDs at the beginning of the report. You can also use the `read_lib` command with the `-lib_messages` option to specify that the report display specific messages first, such as all noise related messages. The `-html` and `-lib_messages` options are not enabled by default.

To generate a library report in HTML format, use the `read_lib` command with the `-html` option, as shown:

```
read_lib lib_file_name -html
```

To generate a report with specific messages first, such as all noise-related messages, use the `read_lib` command with the `-lib_messages` option, as shown:

```
read_lib -html -lib_messages {$read_lib_ccs_noise_msg} lib_file_name
```

where `read_lib_ccs_noise_msg` is a predefined Tcl variable that specifies a list of noise-related messages.

Note:

Currently, the Library Compiler tool provides a predefined list for CCS noise screening results only. Predefined lists for timing and power screening is not available.

After you run either command, the Library Compiler tool generates an `index.html` file. Open the file in a Web browser to display the report. [Figure 2-1](#) shows an example HTML report.

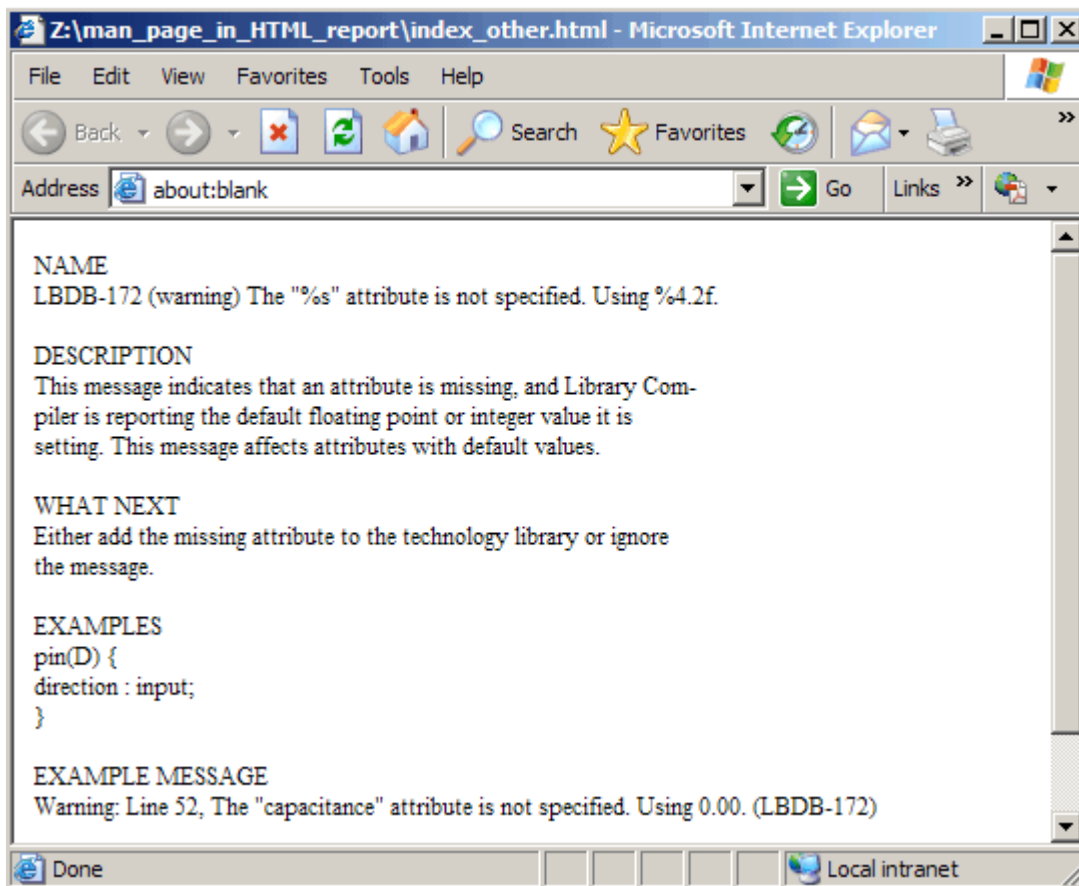
Figure 2-1 read_lib Report in HTML Format

Summary (Total messages: 130 warnings)			
Message Id	Severity (Occurance)	Title	Next Steps To Reduce Severity
LBDB-172	Warning (1)	The '%s' attribute is not specified. Using %4.2f.	Either add the missing attribute to the technology library or ignore the message.
LBDB-272	Warning (8)	The '%s' attribute has a '%f' value, which is less than '%f' the minimum recommended value of th...	Check the library source file, and make sure the value is correct. Refer to ...
LBDB-1054	Warning (121)	The port '%s' does not have the attribute '%s' specified. The value (%f, %f) will be assigned to ...	If you accept the value assigned to the attribute referenced in the error messa...
Detail			
Warning: Line 6, The 'default_cell_leakage_power' attribute is not specified. Using 0.00. (LBDB-172) [link to log file]			
Warning: Line 1416, Cell 'DFFLCRX1', pin 'q', The 'values' attribute has a '-0.004636' value, which is less than '0.000000' the minimum recommended value of this attribute. (LBDB-272) [link to log file]			

Click a message ID, such as LBDB-172 in [Figure 2-1](#), to open the man page.

[Figure 2-2](#) shows the LBDB-172 man page that is linked to the HTML report.

Figure 2-2 Man Page Linked to HTML Report



A “link to log file” link appears after each error message in the HTML report. You can click a “link to log file” link to see the error message displayed in the log file. Figure 2-3 shows a log file that is linked to the HTML report in Figure 2-4 on page 2-64.

Figure 2-3 Log File Linked to HTML Report

```

187: Warning: Line 382, Cell 'DFFLCRX1', Cell 'DFFLCRX1', there is no CCS Noise information. (LBDB-899)
188: Warning: Line 1649, Cell 'DFFLCRX2', Cell 'DFFLCRX2', there is no CCS Noise information. (LBDB-899)
189: Warning: Line 2876, Cell 'DFFLCRX4', Cell 'DFFLCRX4', there is no CCS Noise information. (LBDB-899)
190: Warning: Line 4103, Cell 'DFFLCRX8', Cell 'DFFLCRX8', there is no CCS Noise information. (LBDB-899)
191: Warning: Line 5320, Cell 'DFFRX1', Cell 'DFFRX1', there is no CCS Noise information. (LBDB-899)
192: Warning: Line 5921, Cell 'DFFRX2', Cell 'DFFRX2', there is no CCS Noise information. (LBDB-899)
193: Warning: Line 6512, Cell 'DFFRX4', Cell 'DFFRX4', there is no CCS Noise information. (LBDB-899)
194: Warning: Line 7103, Cell 'DFFRX8', Cell 'DFFRX8', there is no CCS Noise information. (LBDB-899)

```

Prioritizing and Customizing Messages

To control which messages appear on the front page of the report, specify a user-defined variable, such as `message_id`, before you run the `read_lib` command:

```
set message_id {msg_list}
```

For example, if you want the LBDB-644 and LBDB-615 messages to be listed on the first page of the report, set the following:

```
set message_id "LBDB-644 LBDB-615"
```

The Library Compiler tool automatically uses the title from the NAME section of the man page as the title for the message in the report. You can customize the message title in the report to make it more meaningful by specifying a title in curly brackets (`{ }`), as shown:

```
set message_id "{LBDB-899 {This is a check about CCS Noise Model}}"
```

You can also set `read_lib_ccs_noise_msg` and `message_id` together with the `-lib_messages` option, as shown:

```
read_lib -html -lib_messages {$read_lib_ccs_noise_msg $message_id} lib_file_name
```

Figure 2-4 shows the report that is generated after using `read_lib_ccs_noise_msg` to customize the message title and setting the user-defined variable to specify that LBDB-899 be listed on the first page.

Figure 2-4 *read_lib* Report With Customized Title

Summary (Total messages: 0 errors, 138 warnings)			
Message Id	Severity (Occurance)	Title	Next Steps To Reduce Severity
LBDB-899	Warning (80)	This is a check about CCS Noise Model	This is only a warning message. No action is required. If CCS Noise information i...
Other messages: 130 warnings			
Detail			
Warning: Line 382, Cell 'DFFLCRX1', Cell 'DFFLCRX1', there is no CCS Noise information. (LBDB-899) [link to log file]			
Warning: Line 1649, Cell 'DFFLCRX2', Cell 'DFFLCRX2', there is no CCS Noise information. (LBDB-899) [link to log file]			
Warning: Line 2876, Cell 'DFFLCRX4', Cell 'DFFLCRX4', there is no CCS Noise information. (LBDB-899) [link to log file]			
Warning: Line 4103, Cell 'DFFLCRX8', Cell 'DFFLCRX8', there is no CCS Noise information. (LBDB-899) [link to log file]			

3

Library Checking

Advanced IC design relies on high-quality libraries. Ensuring that library data is correct and consistent is an essential step before design creation. The `check_library` command checks logic libraries and physical libraries to facilitate design and verification.

Note:

Although the `check_library` command provides consistency checks, you must ensure consistency between your timing libraries.

This chapter includes the following sections:

- [Library Checking Overview](#)
- [Checking Between Logic Libraries](#)
- [Checking Individual Logic Libraries](#)
- [Checking Physical Libraries](#)
- [Checking Between Logic and Physical Libraries](#)
- [check_library Reports](#)

Library Checking Overview

The `check_library` command performs library checking for logic libraries and physical libraries during and after library creation. To ensure that your libraries are consistent, run the `check_library` command before and after processing your design.

The `check_library` command checks:

- The integrity of individual logic libraries
- The integrity of individual physical libraries
- Consistency between logic libraries
- Consistency between physical libraries
- Consistency between logic libraries and physical libraries

Use the `-logic_library_name` option to specify a list of logic libraries (.db files) to be checked for consistency.

If you do not specify the `-logic_library_name` option, the command determines the logic libraries in the following order:

- The libraries specified using the `set_min_library` command. The `check_library` command checks the minimum and maximum libraries as a pair.
- The libraries specified by the `link_library` and `search_path` variables.

The `check_library` command groups the specified libraries by cell set, and checks consistency across all the libraries of each group. The libraries that cannot be grouped or paired are not checked.

If you do not specify these options, the `check_library` command performs default consistency checks between the logic libraries specified by the `link_library` variable.

Use the `-physical_library_name` option to specify a list of physical libraries (.ndm files) to check for consistency.

To cross-check between a logic library and a physical library, specify both library names:

```
check_library -logic_library_name "a.db" -physical_library_name a.ndm
```

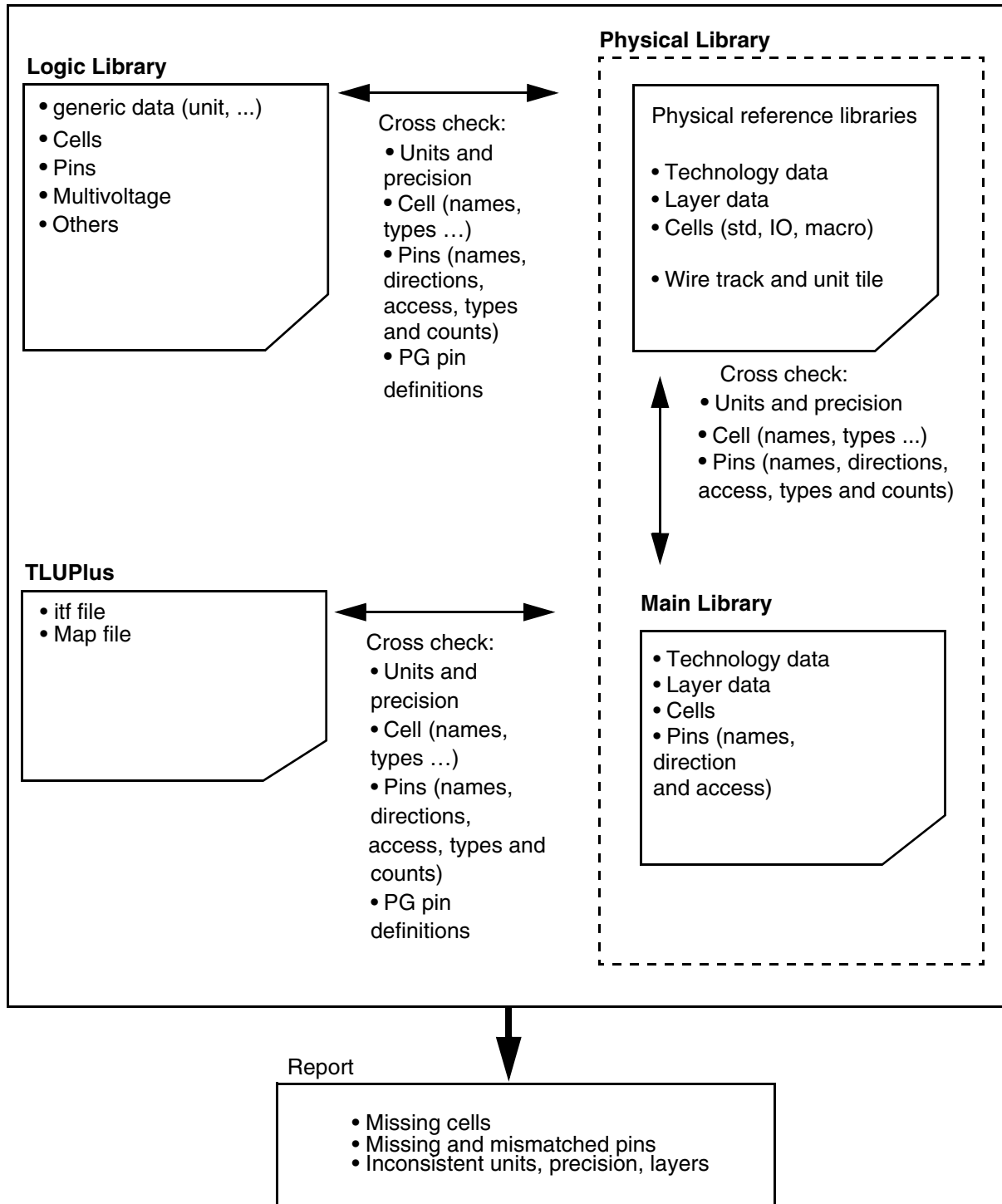
If you do not specify the library names, the command determines the libraries in the following order:

- The logic libraries associated with the physical library (.clib file) opened using the `open_physical_lib` command
- Logic and physical libraries that are already loaded or opened

Use the `-cells` option to specify the list of cell names to check. If you do not use this option, the command checks all the cells in the library.

[Figure 3-1](#) shows an overview of the library checks made by the `check_library` command.

Figure 3-1 Library Checking Overview



Checking Between Logic Libraries

The `check_library` command performs some default as well as specific logic checks on the quality of logic libraries. To enable the specific checks, specify the required options using the `set_check_library_options` command. If you do not specify any options with the `set_check_library_options` command, by default, the `check_library` command performs the general logic checks.

General Logic Checks

By default, the `check_library` command performs the following logic library versus logic library checks.

library Group

At the library level, the `check_library` command checks:

- The library version
- The library type, such as if the library is a power and ground pin type
- Whether the libraries have the same unit
- Whether the default operating conditions are defined

cell Group

At the cell level, the `check_library` command checks if:

- Each library has the same number of cells
- The `area`, `function`, `dont_use`, and `dont_touch` attributes for cells have the same name
- The pin and PG pin names are the same, and the directions, types, functions, and the `voltage_name` attribute values are the same in cells with the same name
- The `input_signal_level` and `output_signal_level` values for rail-based libraries, and the `related_power_pin` and `related_ground_pin` values for PG pin-based libraries are the same, if the attributes are defined
- The order of pins in matching `cell` groups are the same

Pin and PG Pin Groups in the Cells

The `check_library` command checks if the libraries have the same number of pins, including PG pins. The command also checks if the PG pin names, directions, types, functions, and voltage names are the same across libraries. Further, the command checks if the two libraries have the same

- `input_signal_level` or `output_signal_level` values for rail-based libraries and the same `related_power_pin` or `related_ground_pin` values for libraries based on PG pins, if the attributes are defined.
- `pin_number` attribute values; if the values mismatch, the `check_library` command checks the order of pins in each `cell` group, function, and Boolean expression.

Note:

For a library having a level-shifter cell with the `is_unconnected_pg_pin` attribute set to `true` in the `pg_pin` group, the `check_library` command shows an information message, as shown in the following example:

```
Information: The 'VDD' pg_pin is not a 'related_power_pin' pg_pin of
any signal pin on the cell.
```

Timing Arcs

The `check_library` command checks for timing arcs with matching related pins, timing type, timing sense, and `when` conditions across the libraries. These checks are made for any two sets of logic libraries with the same cell group.

Special Logic Checks

To perform special logic library checks, enable the specific checks you want by running the `set_check_library_options` command. [Table 3-1](#) shows the `set_check_library_options` command options that you can use to enable specific logic library checks.

Table 3-1 Special Logic Library Checks

Option	Description
<code>-compare</code>	Checks logic library consistency.
<code>{construct attribute value}</code>	

Table 3-1 *Special Logic Library Checks (Continued)*

Option	Description
<code>-group_attribute</code> <code>{group_name(attribute_name=value)}</code> <code>{group_name/attribute_name=value}</code>	Checks or compares the specified groups or attributes.
<code>-mcmm</code>	Checks consistency for multicornner-multimode.
<code>-scaling {timing noise power}</code>	Checks consistency for CCS timing, CCS noise, and CCS or NLPM power scaling.
<code>-analyze {nominal_vs_sigma table_trend}</code>	Analyzes tables to see how the values change with respect to either slew or load indexes.
<code>-analyze {table_bound table_slope table_index}</code>	Analyzes tables to check for out-of-range absolute values (upper and lower bounds), relative values (value curve slope), or ratios (next index to current index).
<code>-analyze {sensitivity voltage_range}</code>	Analyzes tables to check for CCS noise model sensitivity and output voltage range.
<code>-analyze {value_range}</code>	Analyzes the range of a scalar attribute or table.
<code>-analyze {interpolation}</code>	Checks NLDM index spacing of timing models using interpolation techniques.
<code>-analyze {lvf}</code>	Analyzes Liberty variation format tables for variation density and symmetry.
<code>-analyze {pg_current}</code>	Analyzes ratio of peak current in power and ground <code>pg_current</code> group when output is switching.
<code>-char_integrity</code>	<p>Checks CCS noise accuracy by performing consistency, sensitivity, and voltage range checks together.</p> <p>In the 10 nm mode, also checks NLDM index spacing and Liberty variation format models.</p>

Table 3-1 *Special Logic Library Checks (Continued)*

Option	Description
<code>-criteria</code> <code>{type upper_bound=max_value</code> <code>lower_bound=min_value}</code> <code>{type max_slope=max_value</code> <code>min_slope=min_value}</code> <code>{type</code> <code>max_index_ratio=max_ratio_va</code> <code>lue}</code> <code>{type</code> <code>max_sigma_to_nominal_ratio=m</code> <code>ax_ratio1</code> <code>max_sigma_to_sigma_ratio=max</code> <code>_ratio2}</code> <code>{group/attribute</code> <code>min_value=val1</code> <code>max_value=val2}</code> <code>{min_power_to_ground_current</code> <code>_ratio=ratio_val}</code> <code>{max_ccsp_to_nldm_time_const</code> <code>_ratio=ratio_val}</code> <code>{max_ccsp_to_ccst_time_const</code> <code>_ratio=ratio_val}</code> <code>{linear_resistance</code> <code>min_value=val1</code> <code>max_value=val2}</code>	Specifies range checking criteria for out-of-range table values.
<code>-tolerance {type</code> <code>relative_tolerance</code> <code>absolute_tolerance}</code>	Specifies a relative tolerance and an absolute tolerance to compare characterization values.
<code>-upf</code>	Checks consistency for multivoltage and UPF.
<code>-validate {timing value noise}</code>	Checks consistency of library characterization.
<code>-compensation</code>	Waives off small CCS noise inconsistencies.
<code>-logic</code>	Enables all the logic library checks.
<code>-leq</code>	Checks logical equivalence for <code>when</code> conditions.

Table 3-1 Special Logic Library Checks (Continued)

Option	Description
<code>-by_group_order</code>	Compares libraries by the library group order for groups that are not unique and groups without names.
<code>-single_mode</code>	Checks extracted timing model libraries, each having a mode
<code>-report_format csv html sql</code>	Generates a report in comma-separated values (CSV), HTML, or SQL format.
<code>-reset</code>	Resets the options to the default.
<code>-all</code>	Enables all logic library checks. Note: You must have a Galaxy license to use the <code>-all</code> option. Otherwise, the <code>check_library</code> command issues an error message.

Note:

To enable all the logic library checks, use the `-logic` option.

To enable all the logic library checks, use the `-all` option.

To reset the library checks to default settings, run the `set_check_library_options -reset` command.

The `check_library` special logic checks are described in the following sections:

- [Comparison Checks](#)
- [Checking Specific Groups and Attributes](#)
- [Comparing Libraries by Group Order](#)
- [Maximum Transition and Maximum Capacitance](#)
- [Scaling Checks](#)
- [Multivoltage and UPF](#)
- [Power Optimization Checks](#)
- [Multicorner-Multimode](#)
- [Specifying Tolerances](#)
- [Library Validation](#)

- [Compensation for the PrimeTime Tool](#)
- [Logical Equivalence Checks for when Conditions](#)
- [Inverter and Buffer Cells](#)
- [Library Analysis](#)
- [Checking CCS Consistency, Sensitivity, and Voltage Range Together](#)
- [Checking ETM Libraries](#)
- [Qualifying Model File Libraries](#)

Comparison Checks

When you specify the `-compare` option with the `set_check_library_options` command, the `check_library` command compares two libraries for missing constructs (also called groups), attributes, and values of the attributes. If you specify more than two libraries, the command compares the first two libraries.

If you specify only the `-compare` option with the `set_check_library_options` command to check a single library, the Library Compiler tool generates an error message and stops checking.

If you specify both the `-compare` and one or more of the `-validate`, `-analyze`, and `-upf` options with the `set_check_library_options` command to check a single logic library, the `check_library` command ignores the `-compare` option and performs the other specified checks. The Library Compiler tool generates a warning message to indicate that the `-compare` option is ignored.

Specify the `set_check_library_options` command with the `-compare` option as follows:

```
prompt> set_check_library_options -compare {construct|attribute|value}
```

The arguments are described as follows:

- `construct`
Checks for both missing constructs and attributes.
- `attribute`
Checks for both missing constructs and attributes, and inconsistent attribute values. The argument checks all groups, subgroups, and attribute values except characterization values. The `-compare attribute` option includes the checks of the `-compare construct` option.

- `value`

Checks for both missing constructs and attributes, and inconsistent attribute values including characterization values. The argument compares the values for each group and of each attribute. The `-compare value` option includes the checks of the `-compare attribute` option.

The `check_library` command also compares user-defined groups, attributes, and values, such as effective current source models (ECSM), between logic libraries.

Note:

User-defined groups and attributes are specified by the `define_group` and `define` statements, respectively. For more information about the `define_group` and `define` statements, see the “Using Library Compiler Syntax” chapter of the *Library Compiler User Guide*.

To compare two library databases for user-defined groups, set the `-compare`, `-report_format`, and `-tolerance` options of the `set_check_library_options` command, as follows:

```
prompt> set_check_library_options -compare {value} \
      -report_format {nosplit} -tolerance {capacitance 0.01 0.01}
prompt> check_library -logic_library_name "lib1.db lib2.db"
```

To compare the attributes from specific user-defined groups in specific cells, use the `-group_attribute` option of the `set_check_library_options` command with the `-cells` option of the `check_library` command as follows:

```
prompt> set_check_library_options -compare {value} \
      -group_attribute {cell_rise cell_fall}
prompt> check_library -logic_library_name "lib1.db lib2.db" \
      -cells {cellname}
```

Note:

To compare user-defined groups with the `check_library` command, use Library Compiler F-2011.09-SP2 or later versions.

Checking Specific Groups and Attributes

Use the `set_check_library_options -group_attribute` command to check or compare specific groups and attributes only. You can specify only a group, only an attribute, a group and an attribute together with a value, or a group and an attribute together without a value.

For example,

- The `set_check_library_options -group_attribute {pin}` command checks the `pin` group only

- The `set_check_library_options -group_attribute {direction}` command checks the direction attribute only
- The `set_check_library_options -group_attribute {pin/direction}` command checks the pin direction only
- The `set_check_library_options -group_attribute {pin/direction=input}` command checks the pin direction when it is an input only

Use a forward slash (/) to combine groups and attributes that you specify together. For example, `{pin/direction}` combines the `pin` group and the `direction` attribute in this group. In this case, the command checks the direction in the pin group only.

Use spaces to separate multiple groups, multiple attributes, and multiple group and attribute combinations.

Use curly brackets ({ }) or double quotation marks (" ") to enclose an expression delimited by spaces. For example, `-group_attribute {"pin = A"}` checks only the pin group whose name is A in all cells or in specified cells.

Use the not (!) symbol to specify that the `check_library` command exclude certain groups and attributes from checking. For example, if you use the following command, the `check_library` command checks all groups and attributes except the `ccs_power` group and the `fall_transition` attribute:

```
prompt> set_check_library_options \
        -group_attribute {!ccs_power !fall_transition}
```

To check or compare all timing-related attributes, all power-related attributes, and all noise-related attributes, specify the `timing`, `power`, and `noise` groups. For example, if you use the following command, `check_library` checks all timing, power, and noise related attributes:

```
prompt> set_check_library_options -group_attribute {timing power noise}
```

The following wildcard characters are available for pattern matching:

- An asterisk (*) matches any string, including a null string
- A question mark (?) matches any single character

For example, when you specify the `set_check_library_options -group_attribute {cell_rise cell_fall output_current_*}` command, the `check_library` command checks and validates nonlinear delay model (NLDM) and composite current source (CCS) timing models only.

If you do not specify the `-group_attribute` option, `check_library` checks all groups and attributes that apply to the specified mode.

Analyzing Groups With Specific Attribute Settings

You can also analyze a library group for a specific attribute setting or condition. To specify the attribute and its value, use the `set_check_library_options -group_attribute` command, as shown:

```
prompt> set_check_library_options -group_attribute \
      {group_name(attribute_name=value)}
```

Note:

Do not insert any space before and after the equal sign (=).

The following examples check:

- Non-monotonic trend in timing arcs where `when = "A * B"`

```
lc_shell> set_check_library_options -analyze {table_trend} \
      -criteria {trend!=/} -group_attribute {timing(when="A * B")}
```
- Non-monotonic trend in timing arcs with `when` conditions

```
lc_shell> set_check_library_options -analyze {table_trend} \
      -criteria {trend!=/} -group_attribute {timing(when=*)}
```
- Non-monotonic trend in default timing arcs

```
lc_shell> set_check_library_options -analyze {table_trend} \
      -criteria {trend!=/} -group_attribute {!timing(when=*)}
```
- Composite current source (CCS) versus nonlinear delay model (NLDM) consistency in hold timing arcs

```
lc_shell> set_check_library_options -analyze {timing noise} \
      -group_attribute {timing(timing_type=hold_rising) \
      timing(timing_type=hold_falling)}
```
- CCS versus NLDM consistency in all timing arcs except where the timing type is minimum pulse width

```
lc_shell> set_check_library_options -analyze {timing noise} \
      -group_attribute {!timing(timing_type=min_pulse_width)}
```
- Table bounds for all output pins

```
lc_shell> set_check_library_options -analyze {table_bound} \
      -group_attribute {pin(direction=output)}
```

Comparing Libraries by Group Order

Use the `-by_group_order` option of the `set_check_library_options` command to compare libraries that have the same order of groups. This option is applicable only to groups that do not have unique names, such as `internal_power()` and `leakage_power()`. Groups with unique names, such as `pin(A)`, are compared by the group names (such as `A`).

To compare two libraries where the groups are specified in the same order, use the following syntax:

```
prompt> set_check_library_options -compare {value} -by_group_order
prompt> check_library -logic_library_name "lib_1.db lib_2.db"
```

When you specify the `-by_group_order` option of the `set_check_library_options` command, the `check_library` command compares the groups and reports missing attributes and attribute values for each group.

If you do not specify the `-by_group_order` option, the `check_library` command compares the groups (that do not have unique names) by the names of the simple attributes in the group. Therefore, if one or more attributes in the group are missing or mismatched, the `check_library` command reports that the group is missing rather than the missing or mismatched attributes.

Maximum Transition and Maximum Capacitance

By default, the `check_library` command checks the following maximum transition and maximum capacitance information:

- Checks for the `max_transition` or `max_capacitance` attributes at the pin level of a CCS library.
In CCS libraries, the `max_transition` and `max_capacitance` attributes are specified for input and output pins, respectively, if the corresponding `default_max_transition` and `default_max_capacitance` attributes are not specified.
- If the `max_transition` and `max_capacitance` attributes are not specified, the `check_library` command checks for the `default_max_transition` and `default_max_capacitance` attributes at the library level.
- If the `max_transition`, `max_capacitance`, `default_max_transition`, and `default_max_capacitance` attributes are not specified, the `check_library` command reports the missing attributes.
- If the `max_transition` and `max_capacitance` attributes are specified, the `check_library` command checks their values against the maximum slew and load index in the pin characterization table.

If the `max_transition` and `max_capacitance` values are greater than the maximum slew and load index, the command reports the inconsistent values.

- If the `default_max_transition` and `default_max_capacitance` attributes are specified and the `max_transition` and `max_capacitance` attributes are not, the `check_library` command checks their values against the maximum slew and load index in the library characterization table.

If the `default_max_transition` and `default_max_capacitance` values are larger than the maximum slew and load index, the command reports the inconsistent values.

Scaling Checks

Specify the `-scaling {timing noise power}` option with the `set_check_library_options` command for the `check_library` command to check:

- The consistency of library data in each library of a scaling library group (SLG) for timing, noise, and power.
- The scaling library group.

The scaling checks are described in the following sections.

Checking Library Data Consistency

The `check_library` command checks for the following consistencies in the library, cell, pin, timing, noise, and power group constructs among the libraries within a scaling library group (SLG). The command also identifies missing models.

General Scaling Consistency Checks

When you specify the `set_check_library_options -scaling` command, the `check_library` command performs the following general consistency checks.

library Group

In a scaling library group (SLG),

- All the libraries must be either PG pin based or non-PG pin based.
- All the libraries must have the same units, and the same trip points including the values of the `slew_derate_from_library` attribute.
- All the libraries must have defined default operating conditions.

- The library process must be the same but the voltage and temperature combinations must be different. The `voltage_map` attribute value combinations must be different across corner libraries.
- The single-rail cells and multirail cells must be in different libraries. This check does not apply to the low power cells. All the low power cells (both single-rail and multirail) can be in the same library.

[Example 3-1](#) shows a report example of inconsistent data across libraries.

Example 3-1 Inconsistent Library Data in Library Scaling Report Example

Warning: List of inconsistent library group data (LIBCHK-300)

Library name	lib#1	lib#2	lib#3 ...
slew_upper_threshold_pct_rise	70.0	80.0	70.0
slew_derate_from_library	0.5	0.7	0.5

[Example 3-2](#) shows a report example of non-compliant operating conditions, such as duplicate PVT.

Example 3-2 Non-compliant Operating Conditions in Library Scaling Report Example

lib#1: tlv1.db
lib#2: tlv2.db

...

Warning: List of incompilant operating conditions (LIBCHK-301)

Library name	lib#1	lib#2	lib#3 ...
nom P/V/T	1.0/0.99/125	1.0/1.1/125	1.0/1.1/125
voltage_map	(VDD,0.99)	(VDD,1.1)	(VDD,1.1)
voltage_map	(VSS,0.00)	(VSS,0.00)	(VSS,0.00)
default_opcon	oc_125_0.99	oc_125_1.1	oc_125_1.1

cell Group

Each library in a scaling library group must have:

- Equal number of cells. To belong to the same library group, at least half the cell names must be the same across libraries.
- Same cell names with consistent attributes: `threshold_voltage_group`, `area`, `function`, `dont_use`, and `dont_touch`.
- Same values of the `input_voltage_range` and `output_voltage_range` attributes.
- Same values of the power management attributes, that is, the `is_level_shifter`, `is_isolation_cell`, `retention_cell`, `always_on`, `switch_cell_type`, and `level_shifter_type` attributes.

pin and pg_pin Groups in the Cells of the Same Name

The pin and PG pin groups in the cells of the same name must have:

- Equal number of pins including PG pins.
- Same pin names and corresponding directions, types, functions, and voltage names.
- Same values of the `input_signal_level`, `output_signal_level`, `related_power_pin`, and `related_ground_pin` attributes for PG pin libraries.
- Same derived `pin_number` attribute values.
- Same values of the power management and `pg_pin` group attributes, that is, the `std_cell_main_rail`, `level_shifter_data_pin`, `level_shifter_enable_pin`, `isolation_cell_data_pin`, `isolation_cell_enable_pin`, `retention_pin`, `always_on`, `switch_pin`, `switch_function`, `pg_function`, `related_switch_pin`, `related_pg_pin`, and `related_internal_pg_pin` attributes.
- `max_transition` attribute for input pins and `max_capacitance` attribute for output pins.

Timing Arcs

- The cells with the same name must have same timing arcs with matching related pins, timing type, timing sense, and when conditions.

[Example 3-3](#) shows a report example for some of the libraries with timing arcs missing CCS driver models.

Example 3-3 Missing CCS Driver Timing Arcs Report Example

Warning: List of timing arcs mismatched in logic libraries (LIBCHK-331)

Cell name	pin/ related_pin	when	timing_type	Group/ Attribute	lib#1	lib#2	lib#3
AN2X12	Z/B		combinational	output_current_rise	existing	missing	missing
AN2X12	Z/B		combinational	output_current_fall	missing	missing	existing
AN2X12	A/B		combinational	ccsn_first_stage	missing	missing	existing

Note:

When a model is completely missing from the scaling library group, it is reported only in the `check_library` summary, as shown:

Number of CCS driver models 0

CCS Timing

When you specify the `set_check_library_options -scaling timing` command, the `check_library` command checks if:

- The libraries have the same units and the same slew trip points.
- All the load and slew indexes are the same across the libraries (for setup hold pessimism reduction).
- The receiver models across all scaling libraries exist and are the same type.
- The `when` conditions in the receiver and noise models are in the same order across the libraries.
- The base `curve_x` is the same across the libraries at a number of points and for each value for compact CCS.
- All the libraries have either the compressed or original format for driver data.

The `check_library` command also checks if:

- The number of CCS timing driver model load indexes are the same for each timing arc.
- The output load indexes in each CCS timing driver model timing arc are the same.
- The driver waveform name and indexes in the `normalized_driver_waveform` group are consistent.

However, failing these checks does not generate an error message. The `check_library` command generates one of the following warning messages if these data are inconsistent:

```
Warning: driver_waveform index mismatches found.
```

Or

```
Warning: slew/load index mismatches found.
```

To obtain the mismatch details, compare the libraries.

CCS Noise

Use the `set_check_library_options -scaling noise` command to specify that the `check_library` command checks if the libraries have the same:

- Order of the `when` conditions in the pin models (including receiver models) across the libraries.
- Number of CCS noise models across the libraries for all pins and arcs.
- Simple attributes in the `ccsn_first_stage` and `ccsn_last_stage` groups except for the `millier_cap_rise` and `millier_cap_fall` attributes.

CCS Power

Use the `set_check_library_options -scaling power` command to specify that the `check_library` command checks if the libraries have the same:

- Set of `dynamic_current` and `leakage_current` groups for each cell.
- Set of `intrinsic_parasitic` and `leakage_power` tables for each cell.
- `power_cell_type` attribute for each cell.
- Set of `internal_power` tables for each pin or cell.

For voltage scaling, the `check_library` command checks that the corresponding voltages are within 20 percent. This is not a requirement. The maximum voltage difference is reported in the LIBCHK-304 table in the `check_library` detailed report.

The `check_library` summary shows the number of cells that pass the scaling requirements.

[Example 3-4](#) shows a typical scaling summary report.

Example 3-4 Library Scaling Summary Report Example

```
-----
Number of cells checked           28
Number of cells passed            0
Cell pass rate                   0%

Logic vs. logic library check summary:
Information: Logic library inconsistencies found for timing scaling. (LIBCHK-362)
Information: Logic library inconsistencies found for noise scaling. (LIBCHK-362)
Information: Logic library inconsistencies found for power scaling. (LIBCHK-362)
-----
```

Exceptions to Library Consistency Checking

The `check_library` command checks and reports the following index mismatches in the report summary but does not use them as pass or fail criteria. To obtain the mismatch details, compare the libraries. The command does not check the indexes for exact match.

- The `slew`, `load`, `input_voltage`, and `output_voltage` indexes. For mismatches, the following warning message is generated:

```
Warning: slew/load index mismatches found.
```

- In both noise and timing scaling, the `normalized_driver_waveform` indexes. For mismatches, the following warning message is generated:

```
Warning: driver_waveform index mismatches found.
```

Checking the Scaling Library Group

The `check_library` command performs the following tasks on a scaling library group:

- Checks if the scaling library group is incomplete due to missing voltage or temperature corner libraries.
- Generates a report if the scaling library group contains multiple scaling library groups.
- Checks if a library belongs to multiple scaling library groups.
- Reports if the `link_library` variable setting includes multiple scaling library groups, or does not include any library from the scaling library group.
- Lists the libraries that include single-rail and multirail cells.
- Lists the libraries of the scaling library group that meet the scaling requirements.
- Validates the scaling library group.

[Example 3-5](#) shows a report example for a scaling library group with some corner libraries missing.

Example 3-5 Scaling Library Group With Missing Corner Libraries Report Example

Warning: Missing libraries in scaling groups (LIBCHK-303)

Group name	Temperature	VDD1	VDD2
grp1	-40.0	1.10	1.21

Note:

For multirail libraries, the rail names are listed instead of the voltages.

Based on this report, you can either add the missing libraries to the scaling library group or use a subset of libraries that form a complete scaling group.

[Example 3-6](#) shows a report example with multiple valid scaling library groups listed in the descending order of the number of libraries in the group. The report also shows the temperature (T) and voltage (V) ranges and the maximum voltage difference between the libraries.

Example 3-6 List of Valid Scaling Library Groups Report Example

Information: Valid scaling library groups (LIBCHK-304)

Group name	Library list	Temperature	Voltage	Max delta V
grp1	v1t1.db v2t1.db v1t2.db v2t2.db	-40 to 125	0.9 to 1.2	33.3%
grp2	lib21.db lib12.db lib13.db	-40 to 125	0.9 to 1.2	33.3%

If a scaling library group is not fit for scaling, the `check_library` command generates the LIBCHK-305 error message. This happens when the library data consistency checks fail, the scaling library group has missing corner libraries, one or more corner libraries are removed to create a subset scaling group, or the `link_library` variable setting is not consistent with the scaling library group.

Limitations

The following limitations apply to the scaling checks:

- The `check_library` pass or fail criteria for scaling libraries do not apply to ETM and rail-based libraries. The library consistency checks fail with ETM libraries because ETM libraries do not have CCS data.
- The `check_library` command checks either all the library data, or the specified cells from the libraries when you use the `-cells` option. The command does not provide an option to exclude cells from the scaling check.

Multivoltage and UPF

When you set the `set_check_library_options` command to `-upf`, the `check_library` command checks:

- For different PVT values for different characterizations
- For the existence of NLDM or CCS models under different voltage conditions
- For the existence of the `voltage_map` attribute and `pg_pin` groups in a multiple PG pin-based library with consistent attributes
- For the existence of the `voltage_map` attribute with at least one map for the voltage value 0.0.

For example,

```
voltage_map ( VSS, 0.0 );
```

where VSS is a voltage name. The `check_library` command issues an error message if this requirement is not met.

- Checks to ensure that the `voltage_map` for the nominal voltage is referenced in the `pg_pin` groups. For example, if you have the following `voltage_map` nominal voltage in a `pg_pin(VDDC)` group,

```
voltage_map ( VDD, 1.2 );
```

VDD should be referenced as the voltage name in the `pg_pin` group, as shown in the following example:

```
pg_pin (VDDC) {
    pg_type : primary_power;
    voltage_name : VDD;
}
```

- If the output or inout pins have the same `power_down_function` across the libraries
- For the existence of power data for all operating conditions
- The `power_down_function` is specified on all output or inout pins with the same values
- The `power_down_function` expressions contain the cell's PG pins that are associated with the signal pins and have no output directions
- A signal pin has one `related_power_pin` attribute and one `related_ground_pin` attribute for cells that are not partial PG cells
- If the `always_on` inverter and `always_on` buffer cells are specified
- If the `input_voltage_range` and `output_voltage_range` attributes specified on the same pin have identical values among all PVT libraries
- The cell structure of low-power cells and issues a warning message if the cell is not UPF-compliant
- All the libraries are PG pin based, meaning that the library is based on `pg_pin` syntax
- For the existence of level-shifter, isolation, retention, and switch cells with consistent and complete attributes
- Power management cell attributes for compliance and completeness and reports any missing attributes at the cell, pin, or PG pin-level

UPF Checks for Level-Shifter Cells

When you set the `set_check_library_options` command to `-upf`, the `check_library` command runs the following checks for level-shifter cells:

- Checks that the level-shifter cell has two pins, including one input pin and one output pin or that the level-shifter cell has three pins, including one input pin, one enable pin, and one output pin.
- Checks that the `std_cell_main_rail` attribute is defined on a `primary_power` power pin and that it is specified as the `related_power_pin` value on one of the level-shifter signal pins.
- Checks that the data input and output signal pins are not related to the same power pin.

- Checks that the `input_signal_level` rail name of the data input pin and the `output_signal_level` rail name of the output pin are different.
- Checks that the `level_shifter_enable_pin` pin-level attribute is defined only on an input pin.
- Checks that the `input_voltage_range` and `output_voltage_range` attributes are defined together. For example, if the `input_voltage_range` attribute is defined at the cell level, the `output_voltage_range` attribute should also be defined at the cell level, and if the `input_voltage_range` attribute is specified on the input pin, the `output_voltage_range` attribute should be specified on the output pin.
- Checks that the `input_voltage_range` and `output_voltage_range` lower bound value is less than or equal to the upper bound value.
- Checks that the `input_voltage_range` attribute is set on an input pin, and the `output_voltage_range` attribute is set on an output pin.
- Checks that the `input_voltage_range` and `output_voltage_range` attribute values are consistent with the type of level shifter.
- Checks that each of the input pins define the `input_signal_level` attribute value or the `related_power_pin` and `related_ground_pin` attributes as a pair, or both. Similarly, the output pins must define the `related_power_pin` and `related_ground_pin` attribute values as a pair.
- Checks that the `input_signal_level` attribute is defined on the input pin for overdrive level-shifter and overdrive enable level-shifter cells.
- Checks that the `input_voltage_range` and `output_voltage_range` attributes are defined for overdrive level-shifter and overdrive enable level-shifter cells.
- Checks that the `input_signal_level` and `output_signal_level` attribute rail names are defined in the `voltage_map` attribute.
- Checks that the level-shifter cell's enable pin and output pin are related to the same power pin. You must mark a cell that does not meet this requirement as `dont_touch` and `dont_use`.

UPF Checks for Isolation Cells

When you set the `set_check_library_options` command to `-upf`, the `check_library` command runs the following checks for isolation cells:

- Checks that the `isolation_cell_data_pin` and `isolation_cell_enable_pin` pin-level attributes are specified only on an input pin.
- Checks that isolation cells have three pins, including one input data pin, one enable pin, and one output pin.

- Checks that the `related_power_pin` and `related_ground_pin` attribute information is specified on the input and output pins. An isolation cell must have all the signal pins related to a power pin specified using the `related_power_pin` attribute and all the signal pins related to a ground pin specified using the `related_ground_pin` attribute. This check applies only to multiple-rail cells.
- Checks that there is a backup power or backup ground pin if an isolation cell has two or more power or ground pins.
- Checks that the output pin is related to a backup power through the `related_power_pin` attribute if a backup power pin exists.

UPF Checks for Switch Cells

When you set the `set_check_library_options` command to `-upf`, the `check_library` command runs the following checks for switch cells:

- Checks that the `switch_cell_type` attribute is specified with the `fine_grain` value if the `is_macro_cell` attribute is set to `true` at the cell level.
- Checks that the `switch_function` attribute contains only switch pins whose input pin (set by the `switch_pin` attribute) is set to `true`.
- For a fine-grained cell, the `pg_function` and `switch_function` attributes are defined in a `pg_pin` group where the `direction` attribute is set to `internal`.
- Checks coarse-grain switch cells to ensure that
 - The `switch_cell_type` attribute is specified with the `coarse_grain` value.
 - The `switch_function` attribute is defined to identify the control logic of its switch pins.
 - It has at least one switch pin.
 - It has at least one controlled PG pin and one regular PG pin. That is, it must have a virtual PG pin with a `pg_type` attribute value defined as `internal_ground` and a primary ground PG pin for a footer, or a virtual PG pin with a `pg_type` attribute value specified as `internal_power` and a primary power PG pin for a header.
 - Each of the output PG pins has a `pg_function` Boolean expression containing input PG pins.
 - The `pg_function` attribute contains only the input power and ground pins.
 - Checks that the `pg_function` and `switch_function` attributes are defined in a `pg_pin` group where the `direction` attribute is set to `inout` or `output`.

UPF Checks for Retention Cells

When you set the `set_check_library_options` command to `-upf`, the `check_library` command runs the following checks for retention cells:

- Checks that the `retention_pin` attribute is defined on an input pin.
- Checks that the `retention_cell` and `retention_pin` attributes are defined together in the same cell.
- Checks that the `power_gating_cell` and `power_gating_pin` attributes are not specified.
- Checks that the `retention_pin` and `power_gating_cell` attributes are not defined in the same cell and the `retention_cell` and `power_gating_pin` attributes are not defined in the same cell.
- If the retention cell has two or more power or ground pins, checks that one of them is a backup power or a backup ground pin.
- If the save and restore pins exist, checks that they are related to the backup power or ground, depending on whether backup power or ground is available.
- Checks that all other signal pins are related to the primary PG pin.

UPF Checks for Always-on Cells

When you set the `set_check_library_options` command to `-upf`, the `check_library` command runs the following checks for always-on cells:

- Checks that the cell has a secondary PG pin as a backup. In another words, always-on cells must have a backup power or backup ground pin to relate to its signal pins.
- Checks that always-on pins are related to a backup PG pin.

UPF Checks for PG Pin and Partial PG Pin Cells

Note:

Black box cells with no timing, noise, and power data, or cells without signal pins do not require at least one power and ground pin. These cells are permitted; the `check_library` command does not issue any warning or error message.

When you set the `set_check_library_options` command to `-upf`, the `check_library` command runs the following checks for partial PG pin cells:

- Checks that load cells, which are cells without output pins, have only one PG pin.
- Checks that tied-off cells have only one PG pin. Depending on the `driver_type`, one power pin is required for pull-up cells and one ground pin is required for pull-down cells.

- Checks that at least one `primary_power` PG pin and one `primary_ground` PG pin exist.
- Checks that the old PG pin syntax based on `power_supply` and `voltage_map`, or `rail_connection` and `pg_pin` are not defined together.
- Checks that the `voltage_name` attribute is defined in both the `pg_pin` group and in the `voltage_map` attribute.

UPF Checks for Substrate Bias Cells

When you set the `set_check_library_options` command to `-upf`, the `check_library` command runs the following checks for substrate-bias cells:

- Checks that the `physical_connection` attribute is associated only with the following predefined bias types: `pwell`, `nwell`, `deepnwell`, and `deeppwell`.
- Checks that the `related_bias_pin` attribute in the `pg_pin` and signal pin groups are valid bias PG pins. A signal pin's `related_bias_pin` value must be one of the related bias pins of the signal pin's related power or ground pin.
- Checks that the PG pin's power type is associated with an `nwell` bias PG pin and the PG pin's ground type is associated with a `pwell` bias PG pin.
- Checks that the association between the PG pin and the bias pin in the signal pin is correct. The bias pin's `pg_type` value should be `pwell`, `nwell`, `deepnwell`, or `deeppwell`.

Power Optimization Checks

Use the `set_check_library_options -optimization {power}` command to check the library cells for power-optimization issues, such as missing high threshold-voltage (Vt) cells.

During leakage power optimization, some of the low-Vt cells in a design, are replaced by equivalent high-Vt cells from a target library.

To specify the low-Vt group names, set the `-criteria` option with the `-optimization` option of the `set_check_library_options` command, as follows:

```
prompt> set_check_library_options -optimization {power}
      -criteria {lvth_groups="lvt_name_list"}
```

The `lvt_name_list` value of the `lvth_groups` attribute is a list of low-Vt group names, such as `lvt` and `svt`. When you use the `-optimization` option, you must specify the values of the `lvth_groups` attribute in the `-criteria` option. Otherwise, the Library Compiler tool issues an error message.

Note:

The Power Compiler tool supports the `set_multi_vth_constraint` command to specify the low-Vt groups. The `lvth_groups` attribute is equivalent to the `set_multi_vth_constraint -lvth_groups` command.

The `check_library` command checks for the presence of high-Vt cells that have the same logic function in a logic library as the low-Vt cells. When you specify the `-optimization {power}` option with the `set_check_library_options` command, the `check_library` command performs the following steps:

- Reads the Vt group attributes of the target-library cells.
- Groups the logically-equivalent library cells.

Note:

To group the library cells, Library Compiler excludes the cells with the `dont_use` attribute and unidentified logic functions.

- Identifies the library cells that do not have logically-equivalent high-Vt cells, based on the Vt group attributes of the target-library cells.
- Reports such library cells and their Vt group attributes. If all the logically-equivalent cells have the specified low-Vt, reports the corresponding logic or function groups. If there are no cells with the specified low-Vt, no report table is displayed.

[Example 3-7](#) shows the `check_library` report for missing high-Vt cells.

Example 3-7 *check_library Report for Missing High-Vt Cells*

Number of function groups missing high threshold_voltage_group: 2 (out of 28)

List of function groups missing high threshold_voltage_group cells
(LIBCHK-313)

function	Library name	Example cell name
A*B	lib1	AND1
A+B	lib1	OR1

Multicorner-Multimode

Use the `set_check_library_options -mcmm` command to specify the following checks for a multicorner-multimode flow:

- Checks for different PVT values for different characterizations between two libraries for different operation conditions, that is, libraries with same-name cells that have different nominal PVT values
- Checks for the same cell attributes, such as `dont_use`, `dont_touch`, and black box

- Checks for the same `power_down_function` for output or inout pins
- Checks for the existence of power data for all operating conditions

Specifying Tolerances

Use the `set_check_library_options -tolerance {type rel_tol abs_tol}` command to specify a relative tolerance and an absolute tolerance to compare characterization values, such as delay values.

The options for the `type` variable are `delay`, `delay_ocv`, `delay_ccsn`, `delay_sensitivity`, `delay_interpolation`, `slew`, `slew_ocv`, `slew_ccsn`, `slew_sensitivity`, `slew_interpolation`, `constraint`, `slew_index`, `load_index`, `time`, `power`, `current`, `voltage`, and `capacitance`. If you do not specify the `type` variable, the default is output load capacitance.

Do not include the unit to specify the absolute tolerance. The tool uses the unit from the first library.

The following example specifies tolerances for `time` and `power`:

```
prompt> set_check_library_options -tolerance {time 0.1 0.2 power 0.3 0.4}
```

If you do not specify tolerances for a type, the default tolerances are used. [Table 3-2](#) lists the default tolerances set in compliance with the PrimeTime tool and the library quality assurance system.

Table 3-2 Default Values for Different Tolerance Types

type	Tolerance	
	Relative	Absolute
<code>delay</code>	0.02 (2%)	2 ps
<code>delay_ocv</code>	0.02 (2%)	5 ps
<code>delay_ccsn</code>	0.03 (3%)	3 ps
<code>delay_sensitivity</code>	0.10 (10%)	10 ps
<code>delay_interpolation</code>	0.03 (3%)	3 ps
<code>delay_interpolation</code> for 10 nm mode	0.01 (1%)	1 ps

Table 3-2 Default Values for Different Tolerance Types (Continued)

type	Tolerance	
	Relative	Absolute
slew	0.03 (3%)	3 ps
slew_ocv	0.03 (3%)	7.5 ps
slew_ccsn	0.05 (5%)	5 ps
slew_sensitivity	0.15 (15%)	15 ps
slew_interpolation	0.03 (3%)	3 ps
constraint	0.04 (4%)	0.015 ns
constraint for 10 nm mode	0.01 (1%)	1 ps
load_index	0.01 (1%)	0.001 pF
time	0.04 (4%)	0.015 ns
power	0.04 (4%)	5 pW
current	0.04 (4%)	0.01 μ A
capacitance	0.01 (1%)	0.001 pF
voltage	0.01 (1%)	Calculated from the default relative tolerance and the rail voltage value

To determine whether two libraries are identical (group by group, and attribute by attribute), set the tolerances to smaller values.

Library Validation

The `check_library` command supports validating CCS timing and noise models including compact-CCS noise models, against their corresponding NLDMs. Use the `-validate` option with the `set_check_library_options` command to validate libraries.

Validating Timing

Use the `set_check_library_options -validate {timing}` command, as shown, to check consistency for library characterization:

```
prompt> set_check_library_options -validate {timing}
```

When you specify the `-validate {timing}` option with the `set_check_library_options` command, the `check_library` command enables the following checks for library characterization:

- Checks for missing NLDM and CCS timing models. If all the models of a particular type (for example, NLDM delay, NLDM slew, CCS driver, or CCS receiver) are missing, the model type is reported in the summary report of the `check_library` command. If the models having some of the timing arcs are missing, they are reported both in the summary and detailed report.
- Checks for the same output load index for each individual cell between CCS and NLDMs and the same load indexes for CCS driver and CCS receiver
- Checks for the same number of NLDM delay and slew indexes and values
- Checks that the delay difference between NLDM and CCS timing models are within an acceptable range for both compact CCS and expanded CCS models
- Checks that the `rise_transition` and `fall_transition` slew difference between NLDM and CCS timing models are within an acceptable range for both compact CCS and expanded CCS models

Validating Timing Tables With Different Dimensions

Use the `-validate {timing value}` option with the `set_check_library_options` command to validate the characterization values (specified by the `values` attribute in a `.lib` file) between models of different dimensions (different number of indexes).

For example, to validate a two-dimensional CCS timing table versus a one-dimensional NLDM timing table, the `check_library` command converts the CCS model to NLDM and compares the values. The values of the CCS and the NLDM timing tables are compared for the same index at each index point. The missing index in the NLDM table is not reported. If all the values are within the specified tolerance, the `check_library` command validates the values. Otherwise, a warning message is issued and the differences between the values are reported.

If you do not specify the `value` argument with the `-validate {timing}` option for CCS versus NLDM validation, the `check_library` command converts the CCS model to NLDM and compares the values for the same index at each index point. If the value tables of the NLDM and CCS models have different dimensions, the missing indexes are reported.

Validating CCS Noise Models Against NLDM

Use the `-validate {noise}` option with the `set_check_library_options` command to validate a CCS noise model against its corresponding NLDM.

When you specify the `-validate {noise}` option with the `set_check_library_options` command, the `check_library` command:

- Checks for missing NLDM and CCS noise models. If all the models of a particular type are missing (for example, CCS noise), the model type is listed in the summary report of the `check_library` command. If the models having some of the timing arcs are missing, they are reported both in the summary and detailed report.
- Compares the delay and slew of the CCS noise model to the NLDM delay and slew for each cell. The values are compared at each input slew and output load index point of the NLDM. If the values are within the specified tolerances, the `check_library` command validates the values. Otherwise, the absolute and relative differences with respect to the NLDM values, are reported.

[Example 3-8](#) shows an example of the validation report.

Example 3-8 CCS Noise Model Validation Report Example

List of inconsistent data between CCS noise and NLDM models (LIBCHK-342)

index_1: input_net_transition

index_2: total_output_net_capacitance

```
-----
fast_125
Error
Cell pin/      timing_type  when Group/Attribute (index_1,index_2)
(NLDM) (CCSN) Absolute Relative Type
name related_pin
-----
XOR  Y      combinational !A&B  cell_rise/values (0.003, 0.001)    24.51
25.88 1.37      5.5%      DELAY
-----
```

Statistical information about the CCS noise model validation check is also reported.

[Example 3-9](#) shows an example of the statistical report.

Example 3-9 CCS Noise Model Validation Statistical Report Example

Report of the statistical analysis results of characterization models. (LIBCHK-353)

```
-----
Mean of differences      Std deviation of differences      Number
of grids
Group type      Absolute Relative      Absolute Relative      Max outlier
Pass      Fail
-----
```

Delay	-0.0439943	-0.02%	0.086984	0.04%	-0.369751	775
121						
Slew	-0.0834122	-0.25%	0.440196	0.45%	-3.17905	683
213						

Note:

To specify the tolerances, use the `delay_ccsn` and `slew_ccsn` types with the `set_check_library_options` command.

For more information about specifying tolerances, see [“Specifying Tolerances” on page 3-28](#).

To perform the noise validation check, specify the `set_check_library_options` and `check_library` commands, as shown:

```
prompt> set_check_library_options -validate {noise} \
        -tolerance {delay_ccsn relative_tolerance absolute_tolerance \
                    slew_ccsn relative_tolerance absolute_tolerance } \
        -report_format {csv=csv_dir}
prompt> check_library -logic_library_name "library_name.db"
```

Compensation for the PrimeTime Tool

The `check_library` command validates CCS timing and noise models against NLDM models.

The `check_library` command can ignore small inconsistencies in CCS noise libraries. To enable this compensatory behavior of the `check_library` command, specify the `-compensation` option with the `set_check_library_options` command.

A library cell that fails a check when the `-compensation` option is not specified but passes it when the `-compensation` option is specified, means that the library has quality issues but the PrimeTime tool can use the library with a slight loss of accuracy.

For example, a library cell with a large input slew of 1000 ps and a relatively small load capacitance. If the cell delay is 50 ps, an absolute error of 10 ps means a large relative error of 20 percent. The absolute error (10 ps) is only one percent of the input slew (1000 ps). Also, a critical path typically does not have such a high input slew.

Logical Equivalence Checks for when Conditions

Use the `set_check_library_options -leq` command to check logical equivalence for `when` conditions. For example, if you set the `-leq` option, the following expressions are reported as identical:

```
when : A+B
when : B+A
```

By default, the `-leq` option is not set, meaning that logical equivalence is disabled and string comparisons are used instead.

Inverter and Buffer Cells

The `check_library` command reports the total number of inverter and buffer cells among different libraries. Use the `set_check_library_options -logic` command to specify that the `check_library` command performs this check.

Library Analysis

To analyze logic libraries, specify the `-analyze` option with the `set_check_library_options` command.

To check a specific group or table, you can specify the `-group_attribute` option with the `-analyze` option. You can also analyze the group for a specific attribute setting or condition. For more information, see [“Analyzing Groups With Specific Attribute Settings” on page 3-13](#).

The `check_library` command performs the following analysis:

- [Value Range Analysis of Attributes](#)
- [Trend Analysis for Single Characterization Tables](#)
- [Variation-Aware Analysis](#)
- [Table Bounds, Slope, and Index Analysis](#)
- [Sensitivity and Voltage Range Analysis](#)
- [NLDM Index Spacing Analysis Using Interpolation](#)
- [Liberty Variation Format Analysis](#)
- [CCS Power Analysis](#)

Value Range Analysis of Attributes

For the `check_library` command to analyze if a scalar or lookup-table attribute is within a specified value range or bounds, use the `value_range` argument with the `set_check_library_options -analyze` command, as shown:

```
prompt> set_check_library_options -analyze {value_range} \
        -criteria {group/attribute min_value=val1 max_value=val2}
```

`attribute` is the attribute name, such as `capacitance`, `transition`, `index_1` or `index_2`.

group is an optional group name, such as `pin`. When you specify the group name, the `check_library` command checks only the attribute in the specified group.

You can use the wildcard character (*) to specify the group and attribute names. For example, `*capacitance` applies to all the attributes ending with `capacitance`, such as `rise_capacitance` and `fall_capacitance`. `cell_*` applies to all the groups starting with `cell_`, such as `cell_rise` and `cell_fall`.

`min_value` and `max_value` are the minimum and maximum values of the attribute in library units. When the attribute value is not in this range, it is reported in the `check_library` report. You must specify at least one of `min_value` and `max_value`.

The following examples check:

- Value range of the `cell_rise` table indexes

```
lc_shell> set_check_library_options -analyze {value_range} \
        -criteria {cell_rise/index_1 min_value=0 max_value=0.1} \
        {cell_rise/index_2 min_value=0 max_value=0.1}
```

- transition attribute range

```
lc_shell> set_check_library_options -analyze {value_range} \
        -criteria {transition min_value=0.1 max_value=0.2}
```

- Value range of the `millier_capacitance` and `capacitance` attributes

```
lc_shell> set_check_library_options -analyze {value_range} \
        -criteria {millier_capacitance min_value=0.1 max_value=0.3 \
        capacitance min_value=0.001 max_value=0.6}
```

- Maximum value of the `cell_leakage_power` attribute

```
lc_shell> set_check_library_options -analyze {value_range} \
        -criteria {cell_leakage_power max_value=0.3}
```

- Minimum value of leakage power in the `leakage_power` group

```
lc_shell> set_check_library_options -analyze {value_range} \
        -criteria {leakage_power/value min_value=0.1}
```

Reporting Value Range Violations

The attributes with values that violate the specified values are reported in the LIBCHK-344 table of the `check_library` report.

For example,

```
lc_shell> set_check_library_options -analyze {value_range} \
        -criteria {leakage_power/value min_value=0.1 max_value=0.3 \
        miller_cap_rise min_value=0.2 max_value=0.6}
lc_shell> check_library
```

generates the following report:

Warning: Table of attribute value range analysis (LIBCHK-344)

Cell name	pin/subgroup	timing_type	when	Attribute	value
INV_13	leakage_power		A	value	0.0641147
INV_13	leakage_power		A	value	0.0633378
INV_13	leakage_power		!(A)	value	0.435334
INV_13	leakage_power		!(A)	value	4.63995e-06
INV_13	X/A		combinational	ccsn_first_stage/ miller_cap_rise	0.00341442

Trend Analysis for Single Characterization Tables

The NLDM and NLPM characterization table values typically follow a trend based on the slew and load indexes. For example, the delay values increase monotonically as slew and load indexes increase.

When you specify the `-analyze {table_trend}` option with the `set_check_library_options` command, the `check_library` command analyzes a single characterization table to see how the values change as either the slew or load indexes change.

[Table 3-3](#) describes the possible trends for a characterization table. As shown, a curve might have multiple peaks.

Table 3-3 Single Characterization Table Value Trends

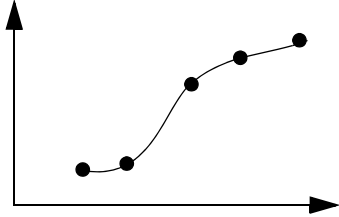
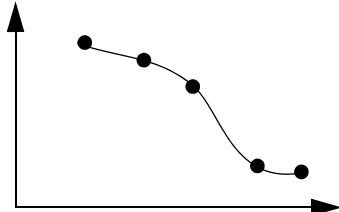
Trend symbol	Trend	Slope
/	Monotonically increasing	
\	Monotonically decreasing	

Table 3-3 Single Characterization Table Value Trends (Continued)

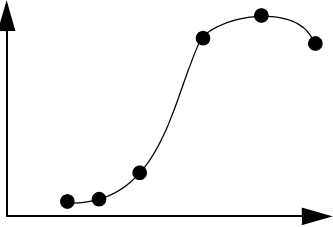
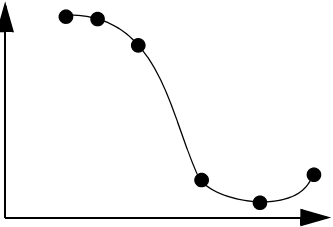
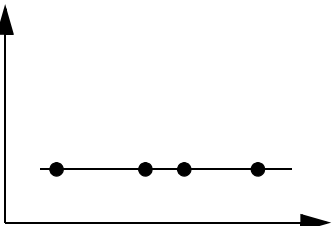
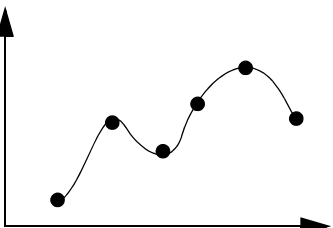
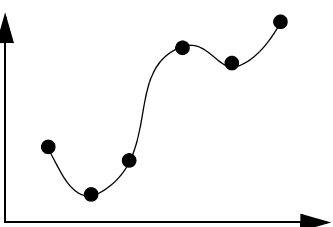
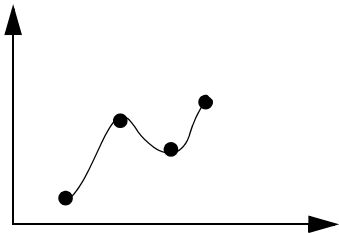
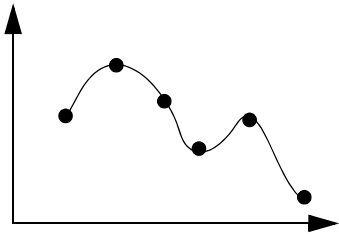
Trend symbol	Trend	Slope
^	Nonmonotonically increasing and then decreasing	
V	Nonmonotonically decreasing and then increasing	
-	Flat	
M	Multiple peaks	
W	Multiple troughs	

Table 3-3 Single Characterization Table Value Trends (Continued)

Trend symbol	Trend	Slope
N	Two peaks where the second peak stays high	
u	Multiple peaks where the second peak does not stay high, that is, decreasing, increasing, and decreasing	
*	Checks all trends	Not applicable

To analyze the trend using any of the trend symbols in [Table 3-3](#), use the `-criteria {trend=trend_symbol}` option with the `-analyze` option.

To specify inequality for a trend, use the exclamation mark followed by an equal sign (!=), such as `{trend!=\\}`. When you specify inequality, the `check_library` command excludes that trend from the analysis.

If you specify the following criteria,

```
prompt> set_check_library_options -analyze {table_trend} \
                                     -criteria {trend=/}
```

the `check_library` command analyzes the values in the table and reports the data that meets the specified criteria, in this case a monotonically increasing trend (/).

You can also specify the `-group_attribute` option to check a specific group. To further narrow the tables to be analyzed, specify the `-cell` option with the `check_library` command. In the following example, the `-group_attribute` option and the `-cell` option specify that the `check_library` command analyze the `fall_power` group in the `ADDER` cell only:

```
prompt> set_check_library_options -analyze {table_trend} \
                                     -criteria {trend!=\\} -group_attribute {fall_power}
prompt> check_library -logic_library_name "lib_typ.db" -cell "ADDER"
```

If you specify the following criteria for an example library, lib.db, that has a cell with the ocv_std_dev_cell_rise lookup tables,

```
prompt> set_check_library_options -analyze {table_trend} \
        -criteria {trend!=/} -group_attribute {ocv_std_dev_cell_rise}
prompt> check_library -logic_library_name "lib.db"
```

the check_library command reports the number of non-monotonically increasing trends in the ocv_std_dev_cell_rise table data and their percentage. This is reported in the check_library summary.

The following summary example shows that there are a total of 1176 trends in the ocv_std_dev_cell_rise tables and only 130 meet the specified criterion of non-monotonically increasing. The text within parentheses are for explanatory purpose and are not part of the summary.

Total number of cells	1	
Number of cells checked	1	(Cells with the specified group conforming to -group option)
Number of cells conformed	1	(Cells meeting the criteria)
Cell conforming rate	100%	
Number of trends checked	1176	(Total number of trends in the specified group)
Number of trends conformed	130	(Number of trends conforming to the specified criteria)
Trend conforming rate	11.0544%	

If you want to specify a monotonically decreasing slope, you must specify the trend in the following format: {trend=\\}

You can also specify absolute and relative tolerance by using the set_check_library_options -analyze {table_trend} command with the -tolerance option. If the table values are within either of the specified tolerances, they are reported as acceptable.

The following example shows that for all constraints, the consecutive table values are acceptable if they are within 2 percent of each other or have an absolute difference of less than 50 ps:

```
prompt> set_check_library_options -group_attribute {*constraint} \
        -analyze {table_trend} -criteria { trend=^ trend=V trend=M } \
        -tolerance {constraint 0.02 0.050}
```

In addition to specifying both absolute and relative tolerance, you can specify only the absolute tolerance or only the relative tolerance. The following example shows that for all constraints, the consecutive table values are acceptable if they are within 50 ps of each other:

```
prompt> set_check_library_options -analyze {table_trend} \
        -tolerance {constraint 0 0.050}
```

The following example shows that for all constraints, the consecutive table values are acceptable if they are within 2 percent of each other:


```
prompt> set_check_library_options -analyze {table_trend} \
      -tolerance {constraint 0.02 0}
```

Checking for 10 nm Logic Libraries

In the 10 nm mode, the `check_library` command analyzes the monotonicity trends of the NLDM `rise_constraint` and `fall_constraint` groups. To enable the 10 nm mode, set the `lc_enable_10nm_mode` variable to `true`.

For example, when you specify the `-analyze {table_trend}` and `-group_attribute {rise_constraint fall_constraint}` options with the `set_check_library_options` command without specifying the `-tolerance` option, the `check_library` command uses the default tolerances of 1 percent and 0.001 ns to analyze the monotonicity trends.

```
prompt> set_check_library_options -analyze {table_trend} \
      -criteria {trend!=/ trend!=\\} \
      -group_attribute {rise_constraint fall_constraint}
```

Variation-Aware Analysis

Variation-aware libraries are characterized using the nominal parameter and the plus and minus sigma variations from the nominal parameter. Therefore, for each pair of input slew and output load indexes, there are two or three delay values.

Instead of analyzing values within a table as described in [“Trend Analysis for Single Characterization Tables” on page 3-35](#), the `check_library` command looks at values across the three characterization tables: at the nominal parameter, the +sigma variation, and the -sigma variation.

When you set the `set_check_library_options` command to the `-analyze {nominal_vs_sigma}` option, the `check_library` command reports the following:

- Monotonicity trends
 - If the delay values increase with the slew and load
 - If the delay values increase or decrease with the variation-aware parameters
- The significance of the variation-aware parameters

If different variation-aware parameters affect delays differently.

The `check_library` command measures the slope of the linear regression of the variation-aware data. If the slope is approximately zero, variation-aware parameters have insignificant impact on the delay and can be ignored to reduce the library size.

Table 3-4 describes the possible trends for variation-aware analysis. As the table indicates, the slope might have three points only, one for the nominal parameter, one for the plus sigma, and one for the minus sigma.

Table 3-4 Trends for Variation-Aware Analysis

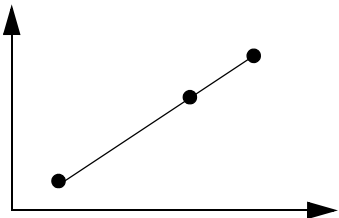
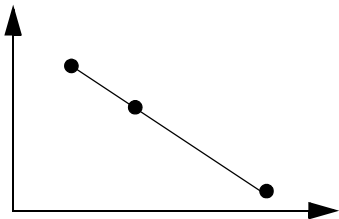
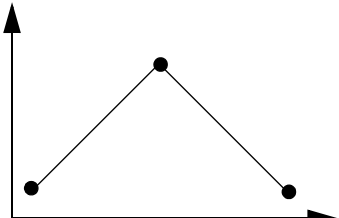
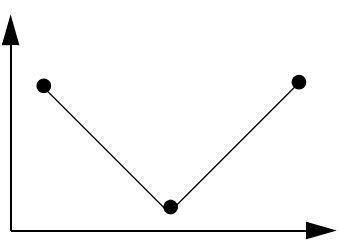
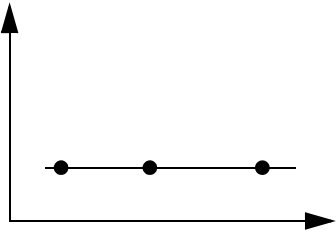
Trend symbol	Trend	Slope
/	If the slope is greater than zero, the trend increases monotonically.	
\	If the slope is less than zero, the trend decreases monotonically.	
^	If the delay nominal parameter is less than the delay at plus sigma and the delay at minus sigma, the trend increases and then decreases.	
V	If the delay nominal parameter is greater than the delay at plus sigma and the delay at minus sigma, the trend increases and then decreases.	

Table 3-4 Trends for Variation-Aware Analysis (Continued)

Trend symbol	Trend	Slope
-	If the slope is approximately zero, the trend is flat (-). So, the impact of the variation-aware parameters on the delay values is insignificant.	
*	Checks all trends	n/a

You can also use the `-criteria` option with the `-analyze` option to specify the comparison criteria for variation-aware analysis. For linear regression models, specify a standard error, slope, and a trend symbol, as shown in the following expression:

```
-criteria {std_error>m.m slope>n.n trend=trend_symbol}
```

where `std_error` is the root mean square deviation of the values from their arithmetic mean. The slope is the sum of products of the deviation of the variation-aware values (x) from the nominal and the deviation of timing values (y) from the nominal, divided by the sum of the square deviations of each variation-aware value from the mean. You can use the greater than ($>$), less than ($<$), or equal ($=$) signs in the expression.

Figure 3-2 shows the slope and the standard error equations.

Figure 3-2 Defining Slope and Standard Error

Linear model:

$$y = \beta_0 + \beta_1 x$$

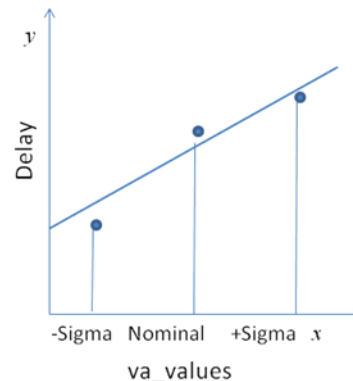
Where x is normalized by dividing `va_values` by `nominal_va_values`

1) Slope

$$\hat{\beta}_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

2) Std error

$$\hat{\sigma}_\epsilon = \sqrt{\frac{SSE}{N-2}} \quad SSE = \sum_{i=1}^N (y_i - \hat{y}_i)^2$$



To analyze only the variation-aware driver models, use the `va_driver` attribute to simplify the `-group_attribute` setting as shown:

```
prompt> set va_driver {"compact_ccs_* va_compact_ccs_*"}
prompt> set_check_library_options -analyze {nominal_vs_sigma} \
    -criteria {"trend="/} -group_attribute {$va_driver}
```

To use specific parameters in the analysis, specify the `va_parameters` attribute as shown:

```
prompt> set_check_library_options -analyze {nominal_vs_sigma} \
    -group_attribute {"va_parameters=(par*)" \
    trend=trend_symbol std_error>m.m slope>n.n}
```

where `par*` is a user-defined parameter. The accuracy of the linear model increases by reducing the standard error. A large slope indicates that the value of the `va_parameters` attribute significantly affects the delay. [Example 3-10](#) shows a variation-aware analysis report.

Example 3-10 Variation-Aware Statistical Analysis Report

Information: Table of variation-aware model analysis (LIBCHK-352)

Cell name: CELL_VAR

va_parameters: (par1, par2, par3, par4, par5, MM)

index_1: related_net_transition

index_2: constrained_net_transition

				values		nominal	-Sigma
pin/	timing_type	when	Group/	va_values	index_1,		
+Sigma	Slope	Std	Trend		index_2		
related_pin			Attribute				
			error				

SI/CP	setup_rising	CDN&D&SE	rise_constraint	MM+/-1	0.005,0.005	0.0723	
0.0703	0.0742	0.004	0.0000	/			
					0.005,0.053	0.0828	
0.0828	0.0868	0.004	0.0009	V			
					0.005,0.438	0.1355	
0.1269	0.1445	0.018	0.0004	/			

Table Bounds, Slope, and Index Analysis

In a library source file, a table consists of characterization values (defined by the `values` attribute) for respective indexes (defined by the `index_i` attribute, where `i` represents the table dimensions). The `check_library` command performs table bounds, slope, and index analysis.

Table Bounds Analysis

The `check_library` command performs the table bounds analysis to check for out-of-bound absolute values in the `values` table. The `check_library` command reports a value that is greater than the specified upper bound or less than the lower bound.

Specify the upper and lower bounds using the `set_check_library_options` command syntax:

```
prompt> set_check_library_options -analyze {table_bound}
        -criteria {type upper_bound=max_value lower_bound=min_value}
        -group_attribute {group_names}
```

type refers to the type of table. [Table 3-6 on page 3-47](#) lists the supported *type* names.

If you specify the `upper_bound` or the `lower_bound` value instead of both the `upper_bound` and `lower_bound` values, the `check_library` command checks for only the specified value. For example, if you have specified only the upper bound value, the command checks for and reports only the values that are greater than the upper bound.

In the following example, the table bounds are specified for the `values` tables of the `output_current_rise` and `output_current_fall` groups:

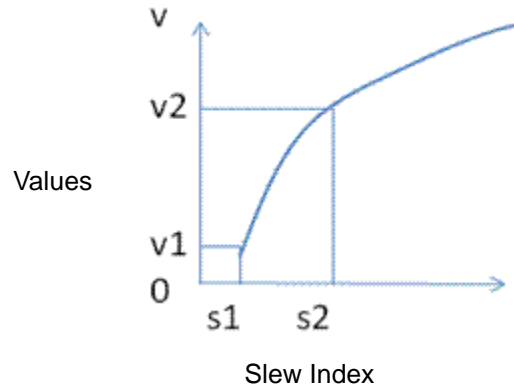
```
prompt> set_check_library_options -analyze {table_bound} \
        -criteria {current upper_bound=2.0 lower_bound=-1.5} \
        -group_attribute {output_current_*}
```

The `check_library` command checks the `values` table for values greater than 2.0 and less than -1.5.

Table Slope Analysis

The `check_library` command performs table slope analysis to check for out-of-range relative values of the table slope. The `check_library` command calculates the slope for two adjacent index points. The slope is normalized to calculate the relative value. Normalization makes the slope independent of the unit and table type. The slope values that are greater than the maximum slope or less than the minimum slope are reported.

For example, for the following slew index versus value curve:



The normalized table slope is:

$$\text{Slope} = \frac{V_2 - V_1}{s_2 - s_1} \times \frac{s_{\text{range}}}{V_{\text{range}}}$$

where,

$$V_{\text{range}} = V_{\text{largest}} - V_{\text{smallest}}$$

$$s_{\text{range}} = s_{\text{largest}} - s_{\text{smallest}}$$

For tables with two indexes, the slope for the first index is calculated at a constant second index point. Then, the slope for the second index is calculated at a constant first index point. For a $m \times n$ table, the slope for $m + n$ curves is calculated. For example, for a 6×7 table, the slope for $6+7=13$ curves, is calculated.

For a linear table curve, the normalized slope is 1. Due to normalization, the slope value is limited to a range or has bounds. [Table 3-5](#) lists the normalized slope range for different types of curves.

Table 3-5 Ranges of Maximum and Minimum Slopes

Curve type	Maximum slope	Minimum slope	Group example
Increasing trend	greater than 1	less than 1	cell_rise or cell_fall
Decreasing trend	less than -1	greater than -1	
All	less than $srange/smin^*$	greater than $-srange/smin^*$	output_current_rise or output_current_fall

*smin is the smallest interval between two index points.

For slope analysis of a table with increasing trend, such as in the `cell_rise` group, specify the maximum slope between 1 to $srange/smin$ and the minimum slope between $-srange/smin$ to 1.

To perform the table slope analysis, use the following syntax:

```
prompt> set_check_library_options -analyze {table_slope}
        -criteria {type/index max_slope=max_value min_slope=min_value}
        -group_attribute {group_names}
```

If you specify only the `max_slope` or the `min_slope` value instead of both the `max_slope` and `min_slope` values, the `check_library` command checks only for the specified value. For example, if you have specified only the minimum slope value, the command checks for and reports only the values that are less than the minimum slope.

`type` refers to the type of table and `index` refers to the type of index. Specifying `index` is optional. If you do not specify `index`, the `check_library` command performs slope analysis for all the indexes of the table. [Table 3-6 on page 3-47](#) lists the supported `type`, `index`, and `group_names`.

In the following example, the maximum and minimum slope values are specified for the `values` table in the `cell_rise` group:

```
prompt> set_check_library_options -analyze {table_slope} \
        -criteria {delay max_slope=1 min_slope=-1} \
        -group_attribute {cell_rise}
```

The `check_library` command checks the `values` table for values that have a positive slope greater than 1 or a negative slope less than -1.

In the following example, the maximum and minimum slope values are specified for two index types, `slew_index` and `load_index`, in a `delay` table:

```
prompt> set_check_library_options -analyze {table_slope} \
```

```
-criteria {delay/load_index max_slope=100 min_slope=-0.5} \
-criteria {delay/slew_index max_slope=30 min_slope=0.9} \
-group_attribute {cell_*}
```

Table Index Analysis

The `check_library` command performs table index analysis to check for out-of-range index values. In library characterization tables, the difference between adjacent index points affects the accuracy of processes, such as timing analysis.

To analyze table index accuracy, the `check_library` command calculates the ratios of each pair of adjacent index points (next index point to current index point). The ratios that are greater than a minimum specified ratio, are reported.

Note:

The ratios are reported only if at least one ratio is greater than the minimum specified ratio. Otherwise, the Library Compiler tool reports an error message.

To perform table index analysis, use the following syntax:

```
prompt> set_check_library_options -analyze {table_index} \
-criteria {type/index max_index_ratio=max_ratio_value} \
-group_attribute {group_names}
```

where `max_index_ratio` is the absolute value of the ratio of the next index point to the current index point.

`type` refers to the type of table and `index` refers to the type of index. Specifying `index` is optional. If you do not specify `index`, the `check_library` command performs index analysis for all the indexes of the table. [Table 3-6 on page 3-47](#) lists the supported `type`, `index`, and `group_names`.

In the following example, the `set_check_library_options` command is used to specify the maximum ratio for a pair of adjacent indexes (ratio of the next index to the current index), in the NLDM delay table:

```
prompt> set_check_library_options -analyze {table_index} \
-criteria { delay/slew_index max_index_ratio=2.0 } \
-group_attribute {output_current_*}
```

The `check_library` command checks the values table for at least one `max_index_ratio` (ratio of the next index to the current index) greater than 2.0.

Supported Type, Index, and Group Names

Table 3-6 lists the supported *type*, *index*, and *group* names for table bounds, slope, and index analysis. The *index* name is ignored in the table bounds analysis.

Table 3-6 Supported Type, Index, and Group Names

Type name	Index name	Group name
delay	slew_index	cell_rise
	load_index	cell_fall
transition	slew_index	rise_transition
	load_index	fall_transition
constraint	related_index	rise_constraint
	constrained_index	fall_constraint
energy	slew_index	rise_power
	load_index	fall_power
current	time	output_current_rise
		output_current_fall
capacitance	slew_index	receiver_capacitance1_rise
	load_index	receiver_capacitance1_fall
		receiver_capacitance2_rise
		receiver_capacitance2_fall
dc_current	input_voltage	dc_current
	output_voltage	
ccsn	time	output_voltage_rise
		output_voltage_fall
		propagated_noise_low
		propagated_noise_high
pg_current	time	pq_current

Sensitivity and Voltage Range Analysis

The `check_library` command supports analyzing the accuracy of CCS noise models, including compact-CCS noise models. These checks verify the quality of library characterization data used by timing analysis tools, such as the PrimeTime tool.

These checks include:

- [CCS Noise Model Sensitivity Analysis](#)
- [Output Voltage Range Analysis](#)

CCS Noise Model Sensitivity Analysis

In some libraries, the CCS noise cell models can be sensitive in delay and slew to their input driving waveform.

To enable the sensitivity check, specify the `sensitivity` argument with the `-analyze` option of the `set_check_library_options` command.

Note:

The sensitivity check does not affect the library cell pass rate. It is better to have a library that passes the sensitivity check than a library that does not.

When you specify the `set_check_library_options -analyze {sensitivity}` command, the `check_library` command checks the sensitivity of the CCS noise models for each timing arc of a cell. At each index point of a CCS noise model, the command checks the model response to two inputs: a normal waveform that reaches the rail voltage; and a clamped waveform that does not reach the rail voltage. If the delay or slew of the CCS noise model for the clamped waveform significantly deviates from the CCS noise model for the normal waveform (that is, exceeds the specified tolerances), the library cell is marked as sensitive and reported. [Example 3-11](#) shows an example of the sensitivity report.

Example 3-11 CCS Noise Model Sensitivity Report Example

```
List of sensitive CCSN models (LIBCHK-343)
index_1: input_net_transition
```

```
-----
Error                                                                 fast_125c_0p99v
Cell name pin/related_pin timing_type  when  Group/Attribute
index_1 (Normal) (Clamped) Absolute Relative Type
-----
XOR3X05  Y      combinational !A&B  cell_rise
0.003    24.51   25.88    1.37    5.5%    DELAY
-----
-----
```

Statistical information is also reported. [Example 3-12](#) shows an example of the sensitivity statistics report.

Example 3-12 CCS Noise Model Sensitivity Statistics Report Example

```
Report of the statistical analysis results of characterization models. (LIBCHK-353)
```

```
-----
Mean of differences  Std deviation of differences
Number of grids
```

Group type	Absolute	Relative	Absolute	Relative	Max outlier	Pass
Fail						
Delay Sensitivity	-0.0439943	-0.02%	0.086984	0.04%	-0.369751	775
121						
Slew Sensitivity	-0.0834122	-0.25%	0.440196	0.45%	-3.17905	683
213						

You can control the clamped waveform voltage by specifying a clamping ratio using the `clamping_ratio` argument with the `-criteria` option of the `set_check_library_options` command. The default clamping ratio is 0.95—that is, the clamped waveform reaches up to 95 percent of the cell rail voltage.

Note:

Specify the tolerances using the `delay_sensitivity` and `slew_sensitivity` types with the `set_check_library_options` command.

For more information about specifying the tolerances, see [“Specifying Tolerances” on page 3-28](#).

To check the sensitivity of a CCS noise model, specify the `set_check_library_options` and `check_library` commands, as shown:

```
prompt> set_check_library_options -analyze {sensitivity} \
      -criteria {clamping_ratio=clamp_val} \
      -tolerance {delay_sensitivity rel_tolerance abs_tolerance \
                  slew_sensitivity rel_tolerance abs_tolerance }
prompt> check_library -logic_library_name "library_name.db"
```

Output Voltage Range Analysis

To check the output voltage range of CCS noise models, use the `voltage_range` argument with the `-analyze` option of the `set_check_library_options` command, as shown:

```
prompt> set_check_library_options -analyze {voltage_range}
prompt> check_library -logic_library_name "library_name.db"
```

When you specify the `voltage_range` argument, the `check_library` command checks the output voltage range of a CCS noise model using the `output_voltage_rise` and `output_voltage_fall` groups defined in the `ccsn_first_stage` and `ccsn_last_stage` groups. The `check_library` command checks if the output voltage range is within the specified tolerances and generates a detailed report for each case where the voltage range of the CCS model is not within the specified tolerances. [Example 3-13](#) shows a typical voltage range report.

Example 3-13 CCS Model Voltage Range Report

Warning: Table of partial voltage ranges on timing arcs (LIBCHK-359)

Cell name	pin/related_pin	timing_type	when	Voltage		Error		Type
				Nominal	Swing	Absolute	Relative	

```
-----
XOR1      z/a      combinational !a      1.08      1.032      -0.048      -4.44%      CCSN
-----
```

You can also view these reports in the comma-separated values (CSV) format. You can also view the corresponding line numbers (from the .lib files) in the CSV file.

Statistical information is also reported. The voltage range pass rate is reported in the summary of the report generated by the `check_library` command. [Example 3-14](#) shows a typical voltage range check summary report. [Example 3-15](#) shows a typical detailed statistical report.

Example 3-14 Voltage Range Statistics in the check_library Summary Report

```
Number of voltage ranges passed      1666
Voltage range pass rate              28.2086%
```

Example 3-15 CCS Model Voltage Range Statistical Analysis Report

Information: Report of the statistical analysis results of characterization models.
(LIBCHK-353)

```
-----
Group type      Mean of differences      Std deviation of differences      Number of grids
                  Absolute Relative      Absolute Relative      Max outlier      Pass      Fail
-----
Voltage range  0.004815      0.45%      0.005051 0.47%      0.041185      1666      4240
-----
```

Note:

Specify the tolerances using the `voltage` type with the `set_check_library_options` command.

For more information about specifying the tolerances, see [“Specifying Tolerances” on page 3-28](#).

To check if the output voltage range of the CCS model is within a specified range, specify the `-analyze {voltage_range}` or the `-char_integrity` option with the `set_check_library_options` command and specify the tolerances, as shown:

```
prompt> set_check_library_options -analyze {voltage_range} \
        -tolerance {voltage relative_tolerance absolute_tolerance}
prompt> check_library -logic_library_name "library_name.db"
```

For example, to check if the output voltage swing is at least 98 percent of the rail voltage, VDD=1.2 V, specify the relative tolerance as 2 percent and therefore the absolute tolerance as 0.024 V, as shown:

```
prompt> set_check_library_options -analyze {voltage_range} \
        -tolerance {voltage 0.02 0.024}
```

NLDM Index Spacing Analysis Using Interpolation

In a nonlinear delay model (NLDM) table, timing values are stored at index points separated by a fixed interval. The stored timing values are used to calculate the timing values at other points of an NLDM index.

Based on the tolerances you specify, the `check_library` command can check if this interval is appropriate. The `check_library` command performs this check using interpolation techniques. This check ensures that the calculated timing values are accurate with respect to the tolerances.

Note:

Specify the tolerances using the `delay_interpolation` and `slew_interpolation` types with the `set_check_library_options` command.

For more information about specifying the tolerances, see [“Specifying Tolerances” on page 3-28](#).

When you enable the NLDM index spacing check, the `check_library` command:

- For each pair of adjacent NLDM index points, calculates the timing value at the midpoint of the two adjacent index points using linear interpolation for one-dimensional and bilinear interpolation for two-dimensional NLDM tables.

In a one-dimensional NLDM table, timing values are stored as discrete values at particular index points of input slew or output load. In a two-dimensional NLDM table, discrete timing values are stored at particular index points of input slew and output load. During delay calculation, the timing values at other points are calculated by interpolating these timing values.

- For each pair of adjacent index points, calculates the timing value at the midpoint of the two adjacent index points using cubic interpolation for one-dimensional and bicubic interpolation for two-dimensional NLDM tables.
- Calculates the difference between the linear and cubic or the bilinear and bicubic values.
- Reports the difference if it exceeds the specified tolerances.

For example, for the two-dimensional 7x7 NLDM table shown in [Figure 3-3](#), the `check_library` command calculates the timing values at the midpoint of each index interval by interpolating the timing values at the grid points. Therefore, the `check_library` command makes six checks along each row and six checks along each column—that is, a total of 36 checks.

Figure 3-3 Two-Dimensional 7X7 NLDM Table With Timing Values Interpolated at Index-Interval Midpoints

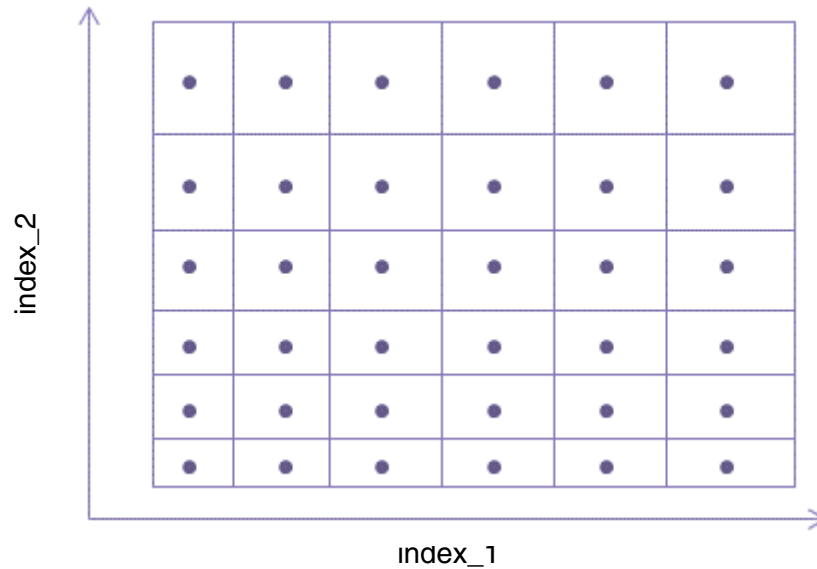
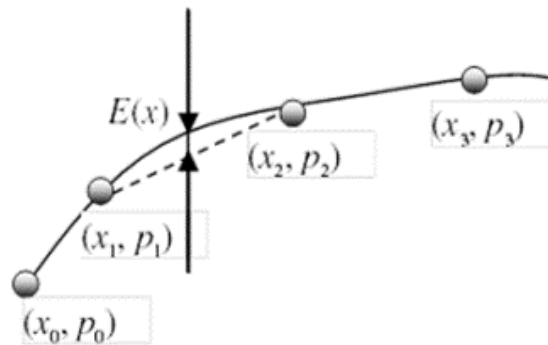


Figure 3-4 A Diagrammatic Representation of the Difference $E(x)$ Between the Bilinear and Bicubic Timing Values



As shown in [Figure 3-4](#), if the difference, $E(x)$, between the bilinear and the bicubic values exceeds with respect to the specified tolerances, it is reported in the `check_library` detailed report with the corresponding timing and index values. [Example 3-16](#) shows a typical index spacing check report.

Example 3-16 NLDM Index Spacing Check Report

Warning: Table of interpolation accuracy analysis (LIBCHK-358)

index_1: related_net_transition

index_2: total_output_net_capacitance

Cell name	pin/ Absolute Relative	timing_type	when	Group	index_1, index_2	Interpolation Error		
						(Bicubic)	(Linear)	
-----	-----	-----	-----	-----	-----	-----	-----	-----
dff	q/clk	rising_edge	!B	cell_rise	1.11,3.71	0.06724	0.07724	0.0100
14.87%								
-----	-----	-----	-----	-----	-----	-----	-----	-----

You can also view these reports in the comma-separated values (CSV) format. You can view the corresponding line numbers (from the .lib files) in the CSV file.

Statistical information is also reported. The interpolation pass rate is reported in the summary of the report. [Example 3-17](#) shows a typical index spacing check summary report. [Example 3-18](#) shows the detailed statistical report.

Example 3-17 NLDM Index Spacing Check Statistics in the check_library Summary Report

Number of interpolations passed	568224
Interpolation pass rate	99.9648%

Example 3-18 NLDM Index Spacing Check Statistical Analysis Report

Information: Report of the statistical analysis results of characterization models.
(LIBCHK-353)

Group type	Mean of differences		Std deviation of differences		Number of grids		
	Absolute	Relative	Absolute	Relative	Max outlier	Pass	Fail
-----	-----	-----	-----	-----	-----	-----	-----
Interpolation	-0.000266	-0.03%	0.001091	4.48%	0.010142	568224	200
-----	-----	-----	-----	-----	-----	-----	-----

To enable the NLDM index spacing check of timing models, use the interpolation argument with the -analyze option of the set_check_library_options command, as shown:

```
prompt> set_check_library_options -analyze {interpolation} \
        -tolerance {delay_interpolation rel_tolerance abs_tolerance \
                    slew_interpolation rel_tolerance abs_tolerance }
prompt> check_library -logic_library_name "library_name.db"
```

For example, to analyze the interpolation accuracy by 4 percent and 0.004 ns for delay and 2 percent and 0.002 ns for slew, specify:

```
prompt> set_check_library_options -analyze {interpolation} \
        -tolerance {delay_interpolation 0.04 0.004 \
                    slew_interpolation 0.02 0.002 }
```

```
prompt> check_library -logic_library_name "my_NLDM.db"
```

Liberty Variation Format Analysis

The Liberty variation format (LVF) syntax consists of groups of random variation of cell delay, output transition, and constraint that are load and input-slew dependent. Each group specifies a lookup table of the variation.

In the lookup table, the absolute variation offsets from the corresponding nominal values are specified at one sigma (σ), where sigma is the standard deviation of the timing distribution. The value of the `sigma_type` attribute specifies if the variation is for an early or a late arrival time.

Table 3-7 shows the Liberty variation format (LVF) and the corresponding nominal timing groups.

Table 3-7 Liberty Variation Format and Nominal Timing Groups

Liberty variation format (LVF) groups	Nominal timing groups
<code>ocv_sigma_cell_rise</code>	<code>cell_rise</code>
<code>ocv_sigma_cell_fall</code>	<code>cell_fall</code>
<code>ocv_sigma_rise_transition</code>	<code>rise_transition</code>
<code>ocv_sigma_fall_transition</code>	<code>fall_transition</code>
<code>ocv_sigma_rise_constraint</code>	<code>rise_constraint</code>
<code>ocv_sigma_fall_constraint</code>	<code>fall_constraint</code>
<code>ocv_sigma_retaining_rise</code>	<code>retaining_rise</code>
<code>ocv_sigma_retaining_fall</code>	<code>retaining_fall</code>
<code>ocv_sigma_retain_rise_slew</code>	<code>retain_rise_slew</code>
<code>ocv_sigma_retain_fall_slew</code>	<code>retain_fall_slew</code>

For more information about the Liberty variation format syntax, see the *Library Compiler User Guide*.

For the `check_library` command to analyze the Liberty variation format data, specify the `lvf` argument with the `-analyze` option of the `set_check_library_options` command as shown:

```
prompt> set_check_library_options -analyze {lvf} \
```



```
-criteria {max_sigma_to_nominal_ratio = max_ratio1 \
          max_sigma_to_sigma_ratio = max_ratio2} \
-group_attribute {group_names}
```

When you specify the `-criteria max_sigma_to_nominal_ratio` option, the `check_library` command calculates the ratios of the variation to the corresponding nominal values and reports the ratios that exceed the specified ratio. These ratios measure the density of the variation distribution. `max_ratio1` is the maximum permissible ratio of the variation to the nominal value.

When you specify the `-criteria max_sigma_to_sigma_ratio` option, the `check_library` command calculates the early to late or the late to early variation values (whichever is greater) and reports the ratios that exceed the specified ratio. These ratios measure the symmetricity of the variation distribution, that is, whether the variation is more for early arrival or late arrival.

The `max_sigma_to_sigma_ratio` argument represents the greater of the early to late and the late to early ratios. `max_ratio2` is the maximum permissible ratio of the early to late and the late to early variation values.

The default `max_ratio1` is 0.25 and the default `max_ratio2` is 3.0.

If at a grid point, the nominal or the one-sigma value of delay or slew is less than or equal to the default absolute tolerance, the `check_library` command skips the LVF analysis for this grid point. To perform the LVF analysis for such grid points, use the `-tolerance` option to specify a tolerance lower than the corresponding default tolerance.

For example, consider the following `.lib` snippet with grid-point values:

```
rise_transition(slew_template) {
  index_1 ("0.0005, 0.01850, 0.03700, 0.14800");
  index_2 ("0.0005, 0.03000, 0.12000, 0.30000");
  values (\
    "0.01005, 0.03500, 0.12700, 0.53000", \
    "0.01005, 0.03500, 0.12700, 0.53000", \
    "0.01005, 0.03500, 0.12700, 0.53000", \
    "0.01005, 0.03500, 0.12700, 0.53000", \
  );
}
ocv_sigma_rise_transition(slew_template) {
  sigma_type : early_and_late;
  index_1 ("0.0005, 0.01850, 0.03700, 0.14800");
  index_2 ("0.0005, 0.03000, 0.12000, 0.30000");
  values (\
    "0.000052, 0.000189, 0.000789, 0.004366", \
    "0.000052, 0.000189, 0.000789, 0.004366", \
    "0.000052, 0.000189, 0.000789, 0.004366", \
    "0.000052, 0.000189, 0.000789, 0.004366", \
  );
}
```

For the following `set_check_library_options` setting, the `check_library` command performs LVF analysis only for the one-sigma value of 0.004366 and the nominal value of 0.53000. The command does not perform the LVF analysis for any of the other values because they are less than the respective absolute default tolerances.

```
prompt> set_check_library_options -analyze {lvf} \
        -criteria {max_sigma_to_nominal_ratio = 0.025 \
                  max_sigma_to_sigma_ratio = 1.25}
```

The default tolerance values are shown in [Table 3-2](#).

To include all the values in the LVF analysis, specify a tolerance less than or equal to the smallest nominal or sigma value in the .lib snippet as shown:

```
prompt> set_check_library_options -analyze {lvf} \
        -criteria {max_sigma_to_nominal_ratio = 0.025 \
                  max_sigma_to_sigma_ratio = 1.25} \
        -tolerance {ocv_slew 0 0.000050 }}
```

You can also analyze the Liberty variation format groups for table monotonicity using the `-analyze {table_trend}` option and for table bounds using the `-analyze {table_bound}` option of the `set_check_library_options` command. For more information about table trend and bound analysis, see [“Trend Analysis for Single Characterization Tables” on page 3-35](#) and [“Table Bounds Analysis” on page 3-43](#).

Liberty Variation Format Analysis Reports

The `check_library` command reports the missing LVF models required for the ratio calculation.

[Example 3-19](#) shows the ratios that do not meet the specified criteria.

Example 3-19 Liberty Variation Format Analysis Report

```
#BEGIN_ANALYZE_LVF
```

```
Information: Table of LVF model analysis. (LIBCHK-352)
```

```
index_1: input_net_transition
```

```
index_2: total_output_net_capacitance
```

Cell pin/ name related_pin	timing_type	when Group	index_1, index_2	Values			Error	
				nominal	early	late	sigma/ nom	sigma/ sigma
AHVT Z/A	combinational	C&!B cell_rise	0.003, 0.0001	0.0279	0.001	0.002	0.065	1.38

```
Number of characterization values meeting the criteria: 12 (out of 392)
```

```
#END_ANALYZE_LVF
```

Example 3-20 shows the arcs for which the nominal nonlinear delay model (NLDM) tables exist but the variation tables are missing.

Example 3-20 Missing Variation Groups Report

Warning: List of timing arcs mismatched in logic libraries (LIBCHK-331)

Cell name	pin/ related_pin	when	timing_type	Group/Attribute	lib#1
HD28	QB/SD	!DS	combinational	ocv_sigma_cell_rise/sigma_type:early	missing

If the variation tables do not exist in the library, it is reported only in the `check_library` summary as shown:

Number of LVF/OCV models 0

To report all the grid points that violate the specified ratios irrespective of the slew or delay tolerances, use the `use -report_all` option with the `check_library` command.

CCS Power Analysis

You can analyze library CCS power models using the `check_library` command as shown in the following sections.

Peak PG Current Ratio Analysis for Switching Outputs

For switching outputs where the `output_switching_condition` attribute is defined in the `switching_group`, most of the current comes from the power supply (rising current) or goes to the ground (falling current). For example,

```
switching_group() {
    ...
    output_switching_condition(...);
    pg_current(VDD) {...}
    pg_current(VSS) {...}
}
```

Therefore, the peak power-to-ground or ground-to-power current ratio is expected to be high.

To check if the peak current (power-to-ground or ground-to-power) ratio of the `pg_current` group under the `switching_group` is within a specified value, specify the `pg_current` argument with the `set_check_library_options -analyze` command. To specify the ratio, use the `min_power_to_ground_current_ratio` argument with the `-criteria` option as follows:

```
lc_shell> set_check_library_options -analyze {pg_current} \
        -criteria {min_power_to_ground_current_ratio=ratio_value}
```

`min_power_to_ground_current_ratio` is the minimum ratio of the power-to-ground or ground-to-power peak currents. The default minimum ratio is 10.

This check applies only to primary power and ground PG pins in the `switching_group` where the `output_switching_condition` attribute is specified.

The `check_library` command reports the peak current ratios outside the specified ratio as shown in the following example.

```
Warning: Table of PG current value ratios (LIBCHK-345)
-----
Cell Group when  Group/          (index_1,      Power   Ground   (P/G)   (G/P)
name            Attribute          index_2)
-----
c1  related_outputs:Z/related_inputs:I/input:rise|output:rise
    !A1 pg_current/peak_value  0.0017,0.00205  1.00252  0.86477  1.159  0.8626
c2  related_outputs:Z/related_inputs:I/input:fall|output:fall
    !A2 pg_current/peak_value  0.0017,0.06806  2.40006  0.514561  4.664  0.2144

Number of PG Current values passed      153
PG Current pass rate                     26.5625%
```

Peak PG Current Bound Analysis

To check if the peak value of the PG current (`pg_current` group under `switching_group`) is greater than 1000 μ A, specify the `-analyze` option with the `set_check_library_options` command as follows:

```
lc_shell> set_check_library_options -analyze {table_bound} \
          -criteria {pg_current upper_bound=1000.0}
```

Time Constant Analysis

To analyze the consistency of the time constant of a CCS power model, specify the `ccsp_time_constant` argument with the `set_check_library_options -analyze` command. For rising outputs, the `check_library` command checks the time constant for primary power. For falling outputs, it checks the time constant for primary ground.

To check if the ratio of the CCS power time constant to the NLDM time constant is within the specified value, use the `max_ccsp_to_nldm_time_const_ratio` argument with the `-criteria` option as follows:

```
lc_shell> set_check_library_options -analyze {ccsp_time_constant} \
          -criteria {max_ccsp_to_nldm_time_const_ratio=ratio_value1}
```

To check if the ratio of the CCS power time constant to the CCS timing time constant is within the specified value, use the `max_ccsp_to_ccst_time_const_ratio` argument with the `-criteria` option as follows:

```
lc_shell> set_check_library_options -analyze {ccsp_time_constant} \
          -criteria {max_ccsp_to_ccst_time_const_ratio=ratio_value2}
```

The default `ratio_value1` or `ratio_value2` is 10.

If the grid points of the NLDM or CCS timing models are not the same as that of the CCS power model, the tool interpolates the NLDM or CCS timing grid points to match the CCS power grid points.

The `check_library` command reports the time constant ratios (CCS power to NLDM or CCS timing) outside the specified ratios as shown in the following example.

Warning: Table of CCSP Time constant value ratios. (LIBCHK-345)

Cell Group name	when Group/Attribute	line_num1	line_num2	(index_1, index_2)	CCSP	NLDM/CCST	Ratio	Type
XOR1	related_outputs:Z/related_inputs:A4/input:rise output:rise							
	!A pg_current/time	10463	40914	0.2109,0.00205	0.714328	0.0254	28.12	CCST
				0.2109,0.06806	0.714328	0.058	12.32	CCST
Number of CCSP Time constant values passed					1032			
CCSP Time constant pass rate					89.5833%			
Number of NLDM vs. CCSP Time constants passed					576			
NLDM vs. CCSP Time constant pass rate					100%			
Number of CCST vs. CCSP Time constant values passed					456			
CCST vs. CCSP Time constant pass rate					79.1667%			

Calculation of Time Constants

In CCS power and CCS timing models, the tool calculates the time constant as the width of the current waveform. From the current-time vector lookup table,

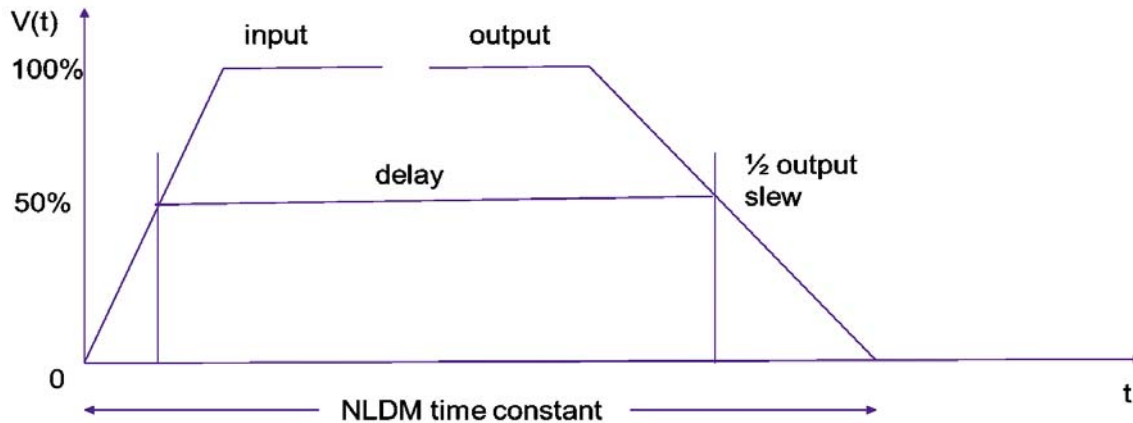
time constant = last time point - first time point

In CCS power models, the current waveforms can have long tails that must be cut to consider the last time point. The tool considers current values below five percent of the peak current as the tail.

The NLDM time constant is calculated from the `input_net_transition`, `rise_transition` or `fall_transition`, and delay values. The `input_net_transition` and `rise_transition` or `fall_transition` values are extrapolated for a complete output swing. The slew threshold and the `slew_derate_from_library` values are also considered.

The following figure shows the calculation of the NLDM time constant for a rising input and falling output.

Figure 3-5 NLDM Time Constant For Rising Input and Falling Output



From Figure , the time constant is:

```
Time constant =
(input_net_transition_full_swing x input_threshold_pct_rise)
+ delay + (fall_transition_full_swing x output_threshold_pct_fall)
```

where

```
input_net_transition_full_swing =
(input_net_transition x slew_derate_from_library)/
(slew_upper_threshold_pct_rise - slew_lower_threshold_pct_rise)
```

```
fall_transition_full_swing =
(fall_transition x slew_derate_from_library)/
(slew_upper_threshold_pct_fall - slew_lower_threshold_pct_fall)
```

Switch Cell Linear Resistance Range Analysis

To check if the linear resistance of a switch cell (calculated from the cell-level `dc_current` group) is within a specified range, use the `set_check_library_options` `-analyze {value_range}` command. To specify the range, use the `linear_resistance` argument with the `-criteria` option as follows:

```
lc_shell> set_check_library_options -analyze {value_range} \
        -criteria {linear_resistance min_value=val1 max_value=val2}
```

`val1` and `val2` are the minimum and maximum linear resistance values in Ohms. The default range is [10, 1000] Ohms.

For example,

```
lc_shell> set_check_library_options -analyze {value_range} \
        -criteria {linear_resistance min_value=10 max_value=1000}
```

Intrinsic Capacitance Range Analysis

To check if the value of the `intrinsic_capacitance` is greater than 0.01 pF, specify the `-analyze` option with the `set_check_library_options` command as follows:

```
lc_shell> set_check_library_options -analyze {value_range} \
        -criteria { intrinsic_capacitance(*)/value max_value=0.010}
```

The `check_library` command reports the `intrinsic_capacitance` value if it is greater than 0.01 pF.

PG Current Range Analysis

To check if the value of the PG current (`pg_current` group under the `leakage_current` group) is in (-1, 1) μ A range, specify the `-analyze` option with the `set_check_library_options` command as follows:

```
lc_shell> set_check_library_options -analyze {value_range} \
        -criteria { pg_current(*)/value max_value=1 min_value=-1} \
        -group_attribute {leakage_current}
```

The `check_library` command reports the `pg_current` value if it is outside the specified range as shown in the following example.

Warning: Table of attribute value range analysis (LIBCHK-344)

Cell pin/ name subgroup	timing_type when	Attribute	value	line_num
XOR1 leakage_current	A1 A2 A3 A4 !Z	pg_current(VDD)/ value	0.0034254	1109

If you use the `report_check_library_options` command, and the `-analyze {value_range}` option is specified with the `set_check_library_options` command, the specified range is reported as shown in the following example.

```
Value range analysis      : true
pg_current(*)/value      : (-0.001, 0.001)
```

Checking CCS Consistency, Sensitivity, and Voltage Range Together

To check CCS library models for consistency, sensitivity, and voltage range together, use the `-char_integrity` option with the `set_check_library_options` command as shown:

```
prompt> set_check_library_options -char_integrity
prompt> check_library -logic_library_name "./lib.db"
```

where the `-char_integrity` option is a single option equivalent to the following individual options specified together:

```
-validate {timing noise} -analyze {sensitivity voltage_range}
```

When you specify the `-validate {timing noise}` option, the `check_library` command performs both the CCS timing and noise validation.

For more information about the CCS timing and noise validation, see [“Library Validation” on page 3-29](#).

When you specify the `-analyze {sensitivity voltage_range}` option, the `check_library` command checks both the sensitivity and the voltage range of the CCS noise models.

For more information about the sensitivity and voltage range checks, see [“Sensitivity and Voltage Range Analysis” on page 3-47](#).

Checking for 10 nm Logic Libraries

When you enable 10 nm logic library checking by setting the `lc_enable_10nm_mode` variable to `true`, the `-char_integrity` option of the `set_check_library_options` command is equivalent to:

```
-validate {timing noise} -analyze {sensitivity voltage_range table_trend interpolation}
```

Checking ETM Libraries

Extracted timing models (ETM) are timing and noise models from gate-level netlists. Each ETM library is modeled for a specific mode, such as read, write, scan, function, and test. An ETM library file can contain a single ETM library or multiple ETM libraries.

Checking ETM libraries include:

- [Qualifying ETM Libraries](#)
- [Grouping ETM Libraries for Specific Checks](#)
- [Reporting ETM Libraries](#)

Qualifying ETM Libraries

The `check_library` command checks the structural consistency of an ETM library and across libraries. These checks are:

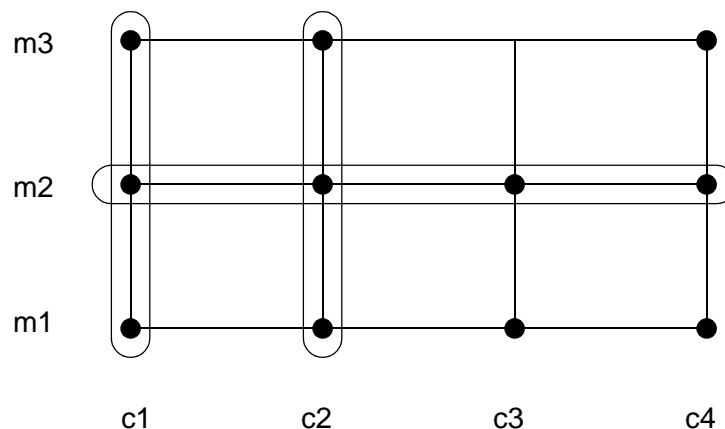
- Each ETM library must have the same process, voltage, and temperature (PVT) or corner and consistent `library`, `cell`, `pin`, and `pg_pin` group level attributes. The ETM libraries with different modes can have different timing arcs.
- For ETM libraries with the same mode but different process, voltage, and temperature (PVT) or corners, the libraries must have consistent structures.

To enable the `check_library` command to qualify ETM libraries, use the `-single_mode` option with the `set_check_library_options` command.

To specify the ETM libraries for qualification, use the `set_lib_group` command. This command allows you to specify the library corners and modes.

Figure 3-6 is a representation of a set of ETM libraries with four corners and three modes.

Figure 3-6 A Set of ETM Libraries With Four Corners and Three Modes



Each dot represents an ETM library. For example, `c1m1.db` is the compiled library file at corner1 and mode1. The 11 ETM libraries are `c1m1.db`, `c1m2.db`, `c1m3.db`, `c2m1.db`, `c2m2.db`, `c2m3.db`, `c3m1.db`, `c3m2.db`, `c4m1.db`, `c4m2.db`, and `c4m3.db`.

The library groups are `grp1/c1`, `grp2/c2`, `grp3/c3` and `grp4/c4`. The `check_library` command checks each library in each of the corners `c1`, `c2`, `c3` and `c4` and checks across the libraries within each corner. The libraries within each corner form a subgroup equivalent to a multiple ETM library file with a particular PVT.

This example shows how to specify the ETM libraries using the `set_lib_group` command.

```
lc_shell> set search_path ". etm_dir"
lc_shell> set_lib_group grp1 -corner c1 -etm_lib \
```

```
lc_shell> set_lib_group grp2 "{c1m1.db m1} {c1m2.db m2} {c1m3.db m3}"
lc_shell> set_lib_group grp3 "{c2m1.db m1} {c2m2.db m2} {c2m3.db m3}"
lc_shell> set_lib_group grp4 "{c3m1.db m1} {c3m2.db m2}"
lc_shell> set_check_library_options -single_mode_etm
lc_shell> check_library
```

Grouping ETM Libraries for Specific Checks

For corner-based checks, that is, multicorner-multimode, UPF, and scaling checks:

- Group the libraries that contain the same cell names.
- In each group with the same cells, further group the libraries with the same process, voltage, and temperature (PVT) or corner.

To specify the group, corner, and library names, use the `set_lib_group` command as shown in the following examples.

[Example 3-21](#) shows corner-based library grouping for multicorner-multimode and UPF checks for the libraries of [Figure 3-6](#). It is optional to specify the ETM libraries with the `link_library` variable.

Example 3-21 Corner-Based Library Grouping for UPF and Multicorner-Multimode Checks

```
lc_shell> set_search_path ". etm_dir"
lc_shell> set_link_library "stdcells.db"
lc_shell> set_lib_group grp1 -corner c1 -etm_lib \
"{c1m1.db m1} {c1m2.db m2} {c1m3.db m3}"
lc_shell> set_lib_group grp2 -corner c2 -etm_lib \
"{c2m1.db m1} {c2m2.db m2} {c2m3.db m3}"
lc_shell> set_lib_group grp3 -corner c3 -etm_lib \
"{c3m1.db m1} {c3m2.db m2}"
lc_shell> set_lib_group grp4 -corner c4 -etm_lib \
"{c4m1.db m1} {c4m2.db m2} {c4m3.db m3}"
lc_shell> set_check_library_options -upf -mcmm
lc_shell> check_library
```

[Example 3-22](#) shows corner-based library grouping for scaling checks for the libraries of [Figure 3-6](#).

Example 3-22 Corner-Based Library Grouping for Scaling Checks

```
lc_shell> set_search_path ". etm_dir"
lc_shell> set_lib_group grp1 -corner c1 -etm_lib \
"{c1m1.db m1} {c1m2.db m2} {c1m3.db m3}"
lc_shell> set_lib_group grp2 -corner c2 -etm_lib \
"{c2m1.db m1} {c2m2.db m2} {c2m3.db m3}"
lc_shell> set_lib_group grp3 -corner c3 -etm_lib \
"{c3m1.db m1} {c3m2.db m2}"
```

```
lc_shell> set_lib_group grp4 -corner c4 -etm_lib \
           "{c4m1.db m1} {c4m2.db m2} {c4m3.db m3}"
lc_shell> set_check_library_options -scaling {timing noise power}
lc_shell> check_library
```

For scaling checks across libraries with the same mode but different corners, use the `create_scaling_lib_group` command instead of the `set_lib_group` command.

[Example 3-23](#) shows mode-based library grouping for scaling checks for the libraries of [Figure 3-6](#). The three library groups are `grp_m1`, `grp_m2`, and `grp_m3`.

Example 3-23 Mode-Based Grouping for Scaling Checks

```
lc_shell> set_search_path ". etm_dir"
lc_shell> create_scaling_lib_group -name grp_m1 \
           "c1m1.db c2m1.db c3m1.db c4m1.db"
lc_shell> create_scaling_lib_group -name grp_m2 \
           "c1m2.db c2m2.db c3m2.db c4m2.db"
lc_shell> create_scaling_lib_group -name grp_m3 \
           "c1m3.db c2m3.db c4m3.db"
lc_shell> set_check_library_options -scaling {timing noise power}
lc_shell> check_library
```

Reporting ETM Libraries

The `check_library` report includes the specified ETM library groups in a table. The following example shows how the ETM library groups are reported.

Library grouping table

Group name	Number	Library name	Library file name
grp1/m1	1	etm_c1m1	/full_path/dir_name/and_lib_file/c1m1.db
grp1/m1	1	etm_c2m1	/full_path/dir_name/and_lib_file/c2m1.db
grp1/m1	1	etm_c3m1	/full_path/dir_name/and_lib_file/c3m1.db
grp1/m1	1	etm_c4m1	/full_path/dir_name/and_lib_file/c4m1.db
grp1/m2	2	etm_c1m2	/full_path/dir_name/and_lib_file/c1m2.db
grp1/m2	2	etm_c2m2	/full_path/dir_name/and_lib_file/c2m2.db
grp1/m2	2	etm_c3m2	/full_path/dir_name/and_lib_file/c3m2.db
grp1/m2	2	etm_c4m2	/full_path/dir_name/and_lib_file/c4m2.db
grp1/m3	3	etm_c1m3	/full_path/dir_name/and_lib_file/c1m3.db
grp1/m3	3	etm_c2m3	/full_path/dir_name/and_lib_file/c2m3.db
grp1/m3	3	etm_c4m3	/full_path/dir_name/and_lib_file/c4m3.db

If you specify multiple library groups with the same group and mode name using the `set_lib_group` command, the `check_library` command does not check the invalid library group setup and reports the number of these groups in the “Number” column.

Qualifying Model File Libraries

During model file compilation, the `read_lib` command skips the screener rule that checks the relationship between the model file and the base library.

To ensure integrity between the model file and the base library, the `check_library` command:

- Checks the library screener rules that were skipped while compiling the model files.
- Checks the consistency of the library units between the model file and the base library databases.
- Checks if the library name, cell names, pin names, and timing arcs match.

The following example shows how to compare the model file library with the base library using the `check_library` command:

```
lc_shell> set_check_library_options -reset
lc_shell> set_check_library_options -compare {value} \
        -tolerance {time 0.00001 0.0000001 \
        slew_index 0.00001 0.0000001 load_index 0.00001 0.0000001}
lc_shell> check_library -logic_library_name "./base.db ./model_file.db"
lc_shell> quit
```

Checking Individual Logic Libraries

For a single library, use the `check_library` command to check the existence of power management cells, specific groups and attributes, and timing, noise, and power models. Based on the options you select, the `check_library` command checks the properties of the library.

If you have both CCS and NLDM timing models in a single library, the `check_library` command validates the following when you specify the `-validate` option:

- The same output load index values for each individual cell between the CCS and NLDM models.
- The same number of NLDM delay and slew indexes and values. For example, it compares two delay indexes.
- The same delay and slew values. It compares the delay and slew values reported in the CCS timing table against the delay and slew values reported in the NLDM table, using the NLDM values as a reference.
- The delay differences between NLDM and CCS timing models within an acceptable range for both compact CCS and expanded CCS models.

- The `rise_transition` and `fall_transition` slew difference between NLDM and CCS timing models within an acceptable range for both compact CCS and expanded CCS models.

Checking Physical Libraries

The `check_library` command checks an individual physical library, and checks multiple physical libraries for consistency. Specify the library names to check using the `-physical_library_name` option with the `check_library` command.

Use the `set_check_library_options` command with the relevant options to specify the checks to perform.

The following table lists the options of the `set_check_library_options` command and the corresponding physical library checks performed by the `check_library` command.

Table 3-8 Physical Library Check Options of the `set_check_library_options` Command

Option	Description
<code>-routeability</code>	Checks pin routeability
<code>-antenna</code>	Checks for missing antenna properties
<code>-signal_em</code>	Checks for missing signal electromigration rules
<code>-same_name_cell</code>	Checks for cells with identical names
<code>-tech</code>	Checks technology file data
<code>-drc</code>	Checks design rules
<code>-placement</code>	Checks cell placement constraints, layers, and sites
<code>-physical</code>	Ease-of-use option to enable all physical library checking options

Checking Between Logic and Physical Libraries

To perform logic library versus physical library checks, specify the commands as shown in the following example:

```
set_check_library_options -logic_vs_physical
report_check_library_options -logic_vs_physical
check_library -physical_library_name a.ndm -logic_library_name "a1.db
a2.db"
```

Default Checks

The `check_library` command performs the following logic library versus physical library checks and reports the missing objects and the libraries where these are missing.

- Missing cells, pins, and PG pins
- Mismatched pins and PG pins
 - When pins or PG pins of the same cell have different attributes (direction, type, or bus) in the logic and physical libraries.
 - When the bus pin and the port type is different between the logic and physical libraries.
 - The checks for mismatched PG types include the `is_secondary_pg` attribute derived from the `.db` file and set on the frame view.

The `check_library` command reports the results of the cross-checking between the logic and the physical libraries in a tabular format. The reports include the counts of the cells and pins with error.

The report begins with `#BEGIN_XCHECK_LIBRARY` and ends with `#END_XCHECK_LIBRARY`. The report for each section begins with `#BEGIN_XCHECK_OPTIONNAME` and ends with `#END_XCHECK_OPTIONNAME`, where `OPTIONNAME` shows the option that you specified with the `set_check_library_options` command.

The header of the report shows the options, tool version, and the check date and time. The library names and the library file names are listed, as shown in the following example.

```
check_library options -logic_vs_physical
Version              O-2018.06
Check date and time  Fri May 26 13:57:25 2018
                    List of logic and physical libraries
-----
Library#   Library name   Library file name
-----
Logic#1    lib1                /dir1/path1/name1/to/lib1.db
Physi#1    lib2                /dir2/path1/name2/phy_libs/lib2.nlib
-----
```

If any inconsistency is detected, the following is printed at the end of the entire report.

```
Information: Logic library is inconsistent with physical library
(LIBCHK-220)
```

If inconsistency is not detected, the following is printed at the end of the report.

Logic vs. physical library consistency checks PASSED.

Missing Cells Report

The `check_library` command reports the missing cells in both logic libraries and physical libraries in two different tables, as shown in the following examples. A physical variant cell is not considered missing if the master cell is detected in the library.

Information: List of cells missing in logic library (LIBCHK-210)

Cell name	Cell type	Physical library
AN2D4	lib_cell	lib2
AN2D8	lib_cell	lib2
AN2D1	lib_cell	lib2

Number of cells missing in logic library: 3 out of 498

Warning: List of cells missing in physical library (LIBCHK-211)

Cell name	Cell type	Logic library
AO2D0	std_cell	lib1
AO2D1	std_cell	lib1

Number of cells missing in physical library: 2 out of 498

In the report, the total number of missing cells is the sum of the missing cells from the logic and the physical libraries.

Missing and Mismatched Pins Report

The missing signal pins and PG pins in logic and physical libraries are reported as shown in the following example. In the `direction` or `direction/type` field, the `direction` attribute value is reported for signal pins and the `pg_type` attribute value is reported for PG pins.

Warning: List of pins missing in logic library (LIBCHK-212)

Physical library	Cell name	Pin name	direction
lib2	AN2D0	A2	in
lib2	AN2D0	VDD	in

Number of cells missing pins in logic library: 1

Warning: List of pins missing in logic library (LIBCHK-212)

Logic library	Cell name	Pin name	direction/type
lib1	BUF1	__VDD	primary power
		__VSS	primary ground
lib1	MUX1	__VDD	primary power
		__VSS	primary ground

Number of cells missing pins in physical library: 2

The mismatched signal pins and PG pins are reported, as shown in the following example. In the Pin direction/type field, the mismatched direction or type is reported. A missing pin is marked as missing in this field.

If only the PG pin direction is not defined in the logic library, the LIBCHK-213w warning message is reported instead of the LIBCHK-213 error.

Error: List of pins mismatched in logic and physical libraries (LIBCHK-213)

Cell name	Pin name	Pin direction/type	
		lib1	lib2
ANTENNA_1	I	signal	diode

Number of cells with mismatched pins: 1

Specific Checks

When you specify the following options with the `set_check_library_options` command, the `check_library` command also performs the specified logic library versus physical library consistency checks.

- `-cell_area`

Cross-checks cell areas between logic and physical libraries

The logic cell area is represented by the `area` attribute in the `cell` group. The physical cell area is measured by the place-and-route boundary. The cell area cross-checking excludes physical-only cells and pad cells.

For consistency, the ratio of the physical-to-logic area of a cell must be within five percent of the average ratio of the physical-to-logic area of all the library cells. The cells with area ratios that violate the average ratio by five percent, are reported.

Warning: List of cells with inconsistent area (LIBCHK-216)

Logic library name	Cell name	Area ratio phys/logic
lib1	AN2D1	1.066
lib1	AN2D2	1.066

Number of cells with inconsistent area: 2

Average cell area ratio physical/logic in the library: 0.9978

- -cell_footprint

Checks the `cell_footprint` attribute value versus the cell place-and-route boundary

This check is performed only when the `cell_footprint` attribute is specified in the `cell` group in a logic library. The cell place-and-route boundary is represented by its coordinates in the physical library. The cells with the same footprint must have the same place-and-route boundaries and the same pin spots.

The cell footprints, associated cell names, and the place-and-route boundary are reported, as shown in the following example. If the place-and-route boundaries are different for the same `cell_footprint` value, the Compatibility field is marked Incompatible.

Warning: List of cells with `cell_footprint` attribute (LIBCHK-215)

Logic library name	cell_footprint	Cell name	PR boundary	Compatibility
lib12	TIEH	TIEH	{0 0}{0.64 2.52}	
lib12	TIEL	TIEL	{0 0}{0.64 2.52}	
lib12	an2d1	AN2D0	{0 0}{1.6 2.52}	Incompatible
lib12	an2d1	AN2D1	{0 0}{1.6 2.52}	Incompatible
lib12	an2d1	AN2D2	{0 0}{1.92 2.52}	Incompatible

Number of incompatible `cell_footprints` in the libraries: 1 out of 3

- -bus_delimiter

Checks bus delimiters in the logic and physical libraries

The bus delimiter is represented by the `bus_naming_style` attribute in the logic library. In the physical library, it is represented by the `BUSBITCHARS` bus-bit characters in the LEF file. You can make the bus delimiters consistent by setting the bus delimiter in the physical library using the `design.bus_delimiters` application option.

The bus delimiters for all the logic and physical libraries are reported.

Information: List of bus naming styles (LIBCHK-214)

Library name	Library type	Bus naming style
lib1	Logic library	Undefined
lib2	Physical library	[]

The bus delimiter inconsistency is reported, as shown:

Inconsistent bus delimiters found in the libraries.

- `-logic_vs_physical`

Ease-of-use option to enable the `-cell_area`, `-cell_footprint`, and `-bus_delimiter` options together

The `-significant_digits`, `-reset`, and `-report_format {nosplit}` options of the `set_check_library_options` command also apply to logic library versus physical library checks. For more information about these options, see the command man page.

Generating Tcl Script to Fix Inconsistencies

You can use the `check_library` command to generate a script to automatically fix inconsistencies between the logic libraries and the physical library. If there are consistency issues, the command generates the script after reporting them. The output script can be sourced in IC Compiler II Library Manager shell (`icc2_lm_shell`).

You can also modify the script to allow missing objects in the `cell`, `pin`, and `pg_pin` groups.

To specify how to fix the consistency issues in the `check_library` generated script, run the `set_lib_mismatch_fix_options` command with the following options before running the `check_library` command, as shown:

```
set_lib_mismatch_fix_options
  -output lib_mismatch_fix_filename.tcl
  -remove_missing { cell | pin | pg_pin }
  -resolve_by_db {db_attribute_list}
  -resolve_by_frame {frame_attribute_list}
```

Use the `-output` option to specify the name of the output script file. The default is `lib_mismatch_fix.tcl`.

Use the `-remove_missing` option to remove the specified element that is missing in the other library. For example, the `-remove_missing {cell}` option removes the cells that exist in either the logic or the physical libraries.

Use the `-resolve_by_db` option to resolve conflicts between the logic (.db) and the physical libraries using the .db attributes as golden. The attributes include `bus_naming_style`, `antenna_diode_type`, `cell_type`, `pin_type`, `pg_type`, and `direction`.

Use the `-resolve_by_frame` option to resolve conflicts between the logic (.db) and the physical libraries using the .frame attributes as golden. The attributes are `bus_naming_style`, `pg_type`, `direction` in the `pg_pin` group.

The following example shows how to check logic libraries versus physical libraries and generate the script to fix the consistency issues:

```
# set options to check consistencies and missing group/attributes
set_check_library_options -logic -logic_vs_physical

# set auto fix options to write lib config file
# if -auto_fix is specified
set_lib_mismatch_fix_options -output lib_auto_fix.tcl
-remove_missing {cell} -resolve_by_db {direction}

# check libraries and write auto fix script
check_library -physical_library a.ndm -logic_library "a1.db a2.db"
```

The following example shows a typical script generated by the `check_library` command.

```
# Auto fix script for test_top
set_current_mismatch_config auto_fix
read_ndm /path/to/test.ndm
# Remove cells with missing pin/pg_pins
# Cells to be removed from physical libraries
set extra_cells "test_top "
foreach lib_cell $extra_cells {
    remove_lib_cell */$lib_cell
}
set_attribute [get_lib_pins */an2*/vdd] direction inout
```

check_library Reports

Use the `report_check_library_options` command to report the values of the options set by the `set_check_library_options` command, as shown:

```
prompt> report_check_library_options -option_name
```

The `report_check_library_options` command has the following options:

`-logic`

Reports the logic library checking options set by the `set_check_library_options` command.

-logic_vs_physical

Reports the logic library versus physical library cross-checking options set by the `set_check_library_options` command.

-default

Reports the default settings for each `check_library` command option. If not specified, the current values are reported.

Report Formats

After checking a library, the `check_library` command generates a report. By default, the report is formatted in tables in a text file. You can also generate the report in the following formats:

- Comma-separated values (CSV) format. For more information, see [“Specifying a Report in CSV Format” on page 3-78](#).
- HTML format. For more information, see [“Specifying a Report in HTML Format” on page 3-81](#).
- SQL format. For more information, see [“Specifying a Report in SQL Format” on page 3-82](#).

Default Text Report

Each text report begins with a summary that reports the library names, locations, units, and specified options:

```
Logic Library #1:
  Library name       : my_design.db
  File name          : path_to_library/design.db
  Library type       : non-pg_pin based db
  Library Version    : Not Specified
  Tool Created       : B-2008.09-SP1
  Data Created       : Fri Apr 18 23:54:52 2008
  Time unit          : lns
  Capacitance unit   : 0.1ff
  Leakage power unit : lpW
  Current unit       : luA
check_library options : -validate { timing }
                      : -group_attribute {cell_*, output_cu*}
                      : -analyze {nominal_vs_sigma }
                      : -criteria {std_error>0.12 slope=0.33 trend=/}
                      : -tolerance {delay 0.04 0.015 slew 0.1 0.03}
Version:              B-2008.09-SP4-CS2-021709
Check date and time:  Sat Feb 21 22:26:10 2009
```

The report starts with a `#BEGIN_CHECK_ITEM` and concludes with an `#END_CHECK_ITEM` string. At the beginning and end of a `check_library` report, the library name and check date are reported as follows:

```
#BEGIN_CHECK_LIBRARY
  Main library name:/mylibs/my_best_cell
  Date and time: Thurs July 10 12:00:00 2008
#END_CHECK_LIBRARY
```

The `check_library` report also provides a summary and statistical analysis for validation and comparison. Each validation section begins with `#BEGIN_XCHECK_LIBVALIDATION` and ends with `#END_XCHECK_LIBVALIDATION`, and each comparison section begins with `#BEGIN_XCHECK_CMP_VALUES` and ends with `#END_XCHECK_CMP_VALUES`.

The report provides a summary that is similar to the following example:

Number of cells checked	1
Number of cells passed	0
Cell pass rate	0%
Number of grid points	2598
Number of grid points passed	2596
Grid point pass rate	99.9230%
Number of delay values passed	875
Delay pass rate	99.8858%
Number of slew values passed	876
Slew pass rate	100%
Number of constraint values passed	910
Constraint pass rate	100%
Number of receiver capacitance values passed	2124
Receiver capacitance pass rate	100%
Number of leakage_power values passed	95
Leakage_power pass rate	96.9388%

Note:

A cell fails a check if the corresponding detailed report shows at least one issue for the cell or its groups—that is, the attributes and groups of the cell are considered in the pass rate calculation. The sensitivity check does not affect the cell pass rate.

The pass rate, r_{cell} is:

$$r_{\text{cell}} = (n_{\text{total}} - n_{\text{fail}}) / n_{\text{total}} * 100\%$$

where n_{total} is the total number of cells checked and n_{fail} is the number of failed cells.

After the summary, the report provides statistical analysis similar to the following example:

Group type	Mean of differences		Std deviation of differences			Number of grids	
	Absolute	Relative	Absolute	Relative	Max outlier	Pass	Fail
Delay	-0.0111607	-0.06%	0.33389	1.82%	-10	895	1
Slew	0	0.00%	0	0.00%	0	896	0
Receiver cap	0	0.00%	0	0.00%	0	920	0
Constraint	0	0.00%	0	0.00%	0	2144	0
leakage_power	102.041	3.80%	1830.15	40.25%	10706.7	95	3
internal_power	8.3169e-09	-62.19%	2.23744	878.98%	-14.5621	324	20

Note:

The Cell pass rate is not 100 percent because the Fail column has non-zero values of Delay, internal_power, and leakage_power.

For examples of several types of reports, see the man pages.

Reporting Missing Groups

The `check_library` command checks multiple libraries and reports missing groups, such as cells. For checks between multiple libraries, only the missing groups are reported. To reduce redundant messages in the `check_library` report, the subgroups, such as pins and the corresponding attributes, are not reported. [Example 3-24](#) shows a `check_library` report for the missing cells in multiple logic libraries.

Example 3-24 The `check_library` Report for Missing Cells

```
#BEGIN_XCHECK_LOGICCELLS
Number of cells missing in library 1: 0 (out of 28)
Number of cells missing in library 2: 12 (out of 16)
Number of cells missing in library 3: 0 (out of 28)
Number of cells missing in library 4: 0 (out of 28)
Information: List of cells missing in logic libraries (LIBCHK-310)
-----
Cell name          lib#1          lib#2          lib#3          lib#4
-----
ISOHID1            existing       missing        existing       existing
LVLHLD4            existing       missing        existing       existing
-----
#END_XCHECK_LOGICCELLS
```

Reporting Line Numbers From Library Source Files

The `check_library` command reports line numbers from the library source (.lib) files. This feature enables you to quickly find the location of the report data and is useful for library analysis, comparison, and validation.

The `check_library` reports have columns to show the line numbers when you compare two .lib files. For each library, the line numbers are reported in a column in addition to the cell,

pin, timing groups and attributes. The line numbers do not appear in the tabular reports when you compare the .db files.

The `check_library` command reports line numbers for:

- Inconsistent CCS noise models
- Inconsistent power models
- Inconsistent data between different timing models
- Variation-aware model analysis
- Characterization timing table trend analysis
- Characterization table bound analysis
- Characterization table slope analysis
- Characterization table index analysis

Note:

When you compare libraries with different number of indexes, such as a CCS library with two indexes versus an NLDM library with a single index, the line numbers for the missing index are reported as 0.

To enable the `check_library` command to report the line numbers, set the following variable to `true` before running the `read_lib` command:

```
lc_shell> set_app_var lc_check_lib_keep_line_number true
```

If you do not correctly specify the variable, or the line numbers are not reported in spite of correctly specifying the variable, the Library Compiler tool issues a warning message that the variable is ignored. The line numbers are reported only when you run the `read_lib` and `check_library` commands in the same session.

To generate the line numbers, use one of the following methods:

- Do not specify the `-logic_library_name` option with the `check_library` command. The `check_library` command checks the libraries loaded by the `read_lib` command. For example,

```
lc_shell> set_app_var lc_check_lib_keep_line_number true
# The libraries being compared must have different names
lc_shell> read_lib lib1.lib
lc_shell> read_lib lib2.lib
lc_shell> set_check_library_options -compare {value}
lc_shell> check_library
```

- Specify the `-logic_library_name` option with the `check_library` command. For example,

```
lc_shell> set_app_var lc_check_lib_keep_line_number true
lc_shell> read_lib cell.lib
lc_shell> read_lib cell_mod.lib
# The specified library names must match the
# library names loaded by the read_lib command
lc_shell> set_check_library_options -compare {value}
lc_shell> check_library -logic_library_name { core.db core_mod.db }
```

Specifying a Report in CSV Format

Use the `set_check_library_options -report_format` command to specify a report in comma-separated values (CSV) format rather than in text format. A report in the CSV format allows you to import the data into a spreadsheet. To specify a report in the CSV format, set the `csv` option, as shown:

```
prompt> set_check_library_options -report_format {csv=csv_dir \
      nosplit sort_by_cell | sort_by_group_type}
```

In the example, `csv_dir` is the user-defined directory where the CSV files are stored. If you do not specify a directory, the CSV files are stored in the current directory.

The name of the report file is a concatenation of the checked library names appended by the check type and the `.csv` extension in the following format:

`lib1_vs_lib2_checktype.csv`

For example, `ff_1.2v_-30c_vs_ss_1.3v_80c_timing_mismatched.csv`.

If the report-file name becomes longer than 255 characters using this format, the tool assigns only the first library name and the checked libraries' count with the `.csv` extension as the report-file name.

Use the `nosplit` option to specify that lines in a report are not split when column fields overflow. Keeping the lines intact allows you to import the data into the spreadsheet more easily.

Use the `sort_by_cell` option to sort the validation and analysis tables by cell name or the `sort_by_group_type` option to sort the validation and analysis tables by the type of group, such as delay, slew, constraint, and receiver capacitance. By default, the tables are sorted by the group type.

You can also specify other options with the `-report_format` option, such as `-validate timing`.

The `check_library` command reports some of the user-defined groups, attributes, and values, such as ECSM models, in the CSV format. User-defined groups and attributes are specified by the `define_group` and `define` statements, respectively.

For more information about the `define_group` and `define` statements, see the “Using Library Compiler Syntax” chapter of the *Library Compiler User Guide*.

The `check_library` command also reports the characterization values for attributes of the cell, pin, timing, power, and noise groups in the CSV format. [Table 3-9](#) shows the attributes that the `check_library` command reports in this format. The report contains the scalar values of these attributes, absolute and relative errors, and corresponding models. [Table 3-10](#) shows the models reported for the timing, power, and noise groups. The `check_library` command generates a separate CSV file for each power model.

Table 3-9 *Characterization Related Attributes Reported by the check_library Command in CSV Format*

Group	Attribute
cell	cell_leakage_power
pin	capacitance
	rise_capacitance
	fall_capacitance
	rise_capacitance_upper
	rise_capacitance_lower
	fall_capacitance_upper
	fall_capacitance_lower
	max_transition
	max_capacitance
timing	min_pulse_width_high
	min_pulse_width_low
	intrinsic_rise
	intrinsic_fall
	rise_resistance
noise	fall_resistance
	miller_cap_rise
	miller_cap_fall

Table 3-9 *Characterization Related Attributes Reported by the check_library Command in CSV Format (Continued)*

Group	Attribute
power	leakage_power
	internal_power
	dynamic_current
	leakage_current
	compact_ccs_power

Table 3-10 *Timing, Power, and Noise Models Reported in the CSV Format*

Group	Attribute	Model
timing	cell_rise, cell_fall	DELAY
	rise_transition, fall_transition	SLEW
	rise_constraint, fall_constraint	CONSTRAINT
	output_current_rise, output_current_fall	CCS_DRV
	compact_ccs_rise, compact_ccs_fall	VA_DRV
	va_compact_ccs_rise, va_compact_ccs_fall	CCS_RCV
	receiver_capacitance, va_receiver_capacitance	VA_RCV
noise	ccsn_first_stage, ccsn_last_stage	CCS_NOISE
power	leakage_power	NLPM_LP
	internal_power	NLPM_IP
	dynamic_current, leakage_current, compact_ccs_power	CCS_POWER

When the data is stored in a CSV file, the text report indicates the information as shown in the following example:

```
Differences in characterization values in cell group saved in file
$lib1_vs_$lib2_cell_value.csv
Differences in characterization values in pin group saved in file
$lib1_vs_$lib2_pin_value.csv
```

The cell and pin CSV reports are stored in the `csv_dir` directory. The timing, power, and noise reports are stored in the `csv_dir/compare_value` directory. The Library Compiler tool creates the `compare_value` directory within the `csv_dir` or current directory when you use the `-compare {value}` option of the `set_check_library_options` command.

Specifying a Report in HTML Format

To view the report of the `check_library` command in the HTML format, specify the following option with the `set_check_library_options` command:

```
prompt> set_check_library_options -report_format {html display}
```

When you specify this option, the `check_library` command launches a Web browser in the background. When the command execution is complete, the browser displays an HTML report.

If you specify the `-report_format` option with the `display` argument, but without the `html` argument:

```
prompt> set_check_library_options -report_format display
```

The Library Compiler tool issues an error message.

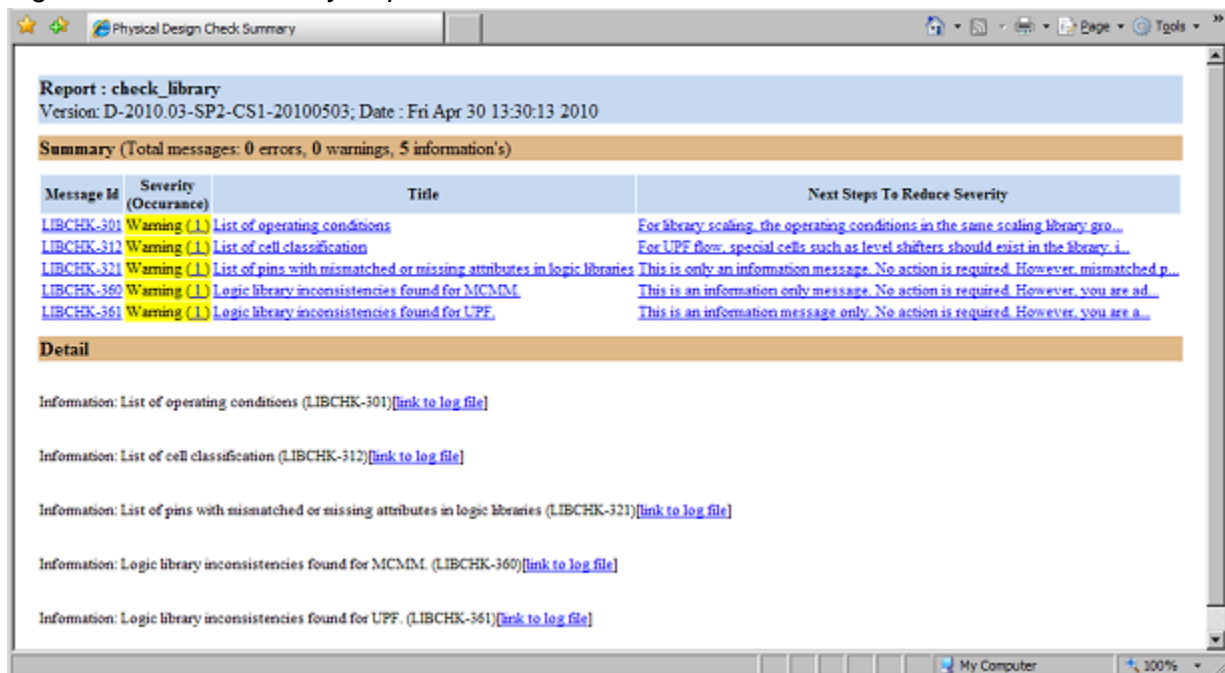
To view the report in the HTML format without launching the browser, specify the following option:

```
prompt> set_check_library_options -report_format { html=html_dir }
```

where `html_dir` is the directory that you specify to store the HTML report files. If you do not specify a directory, the HTML files are stored in the current directory.

Figure 3-7 shows the report in HTML format.

Figure 3-7 *check_library* Report in HTML Format



The Summary section reports the total count of each type of message and includes a table of messages arranged by their severity. For each message, message ID, message severity, message title, and fix suggestion are listed.

Click the message title to view its actual occurrence in the Details section. Click the “link to log file” link to view the associated table or message in the log file. Click the cell name in the table to view expanded message details for the cell.

Click the message ID to view the man page for the message.

Specifying a Report in SQL Format

Use the `set_check_library_options -report_format` command to specify a report in Structured Query Language (SQL) format rather than in text format.

A `check_library` report in the SQL format allows you to load the data in an SQLite browser for analysis or analyze the data using SQL-based scripts. You can analyze the results in the SQLite browser and save the results and plots in the `.csv` or `.jpg` format to view later.

SQLite is a public domain software library that allows you to analyze database files. The benefits include easy querying and visualization windows. This makes it suitable to analyze large data files generated by the `check_library` command.

You can download the SQLite binary or source distribution from:

<http://www.sqlite.org/download.html>

To specify a report in the SQL format, specify the `sql` argument as shown:

```
prompt> set_check_library_options -report_format {sql=database.db3}
```

where `database.db3` is the output report file in the SQL database format.

The following example shows how to generate an SQL output file, `sql_library.db3`, when you check a single library. The `check_library` command runs the characterization integrity checks on the libraries specified by the `link_library` variable. The `check_library` report is obtained as the specified SQL output file.

```
lc_shell> set link_library $target_library
lc_shell> set_check_library_options -char_integrity \
    -report_format {sql=sql_library.db3}
lc_shell> check_library -logic_library_name "$link_library"
```

The following example shows how to generate an SQL output file, `library_field_in.db3` when you check multiple libraries simultaneously. To add the library name field (`lib_name`) to the SQL tables in the `check_library` report, use the `add_field` option.

```
lc_shell> set add_field lib_name # "add_field" is the keyword
lc_shell> set_check_library_options -char_integrity \
    -report_format "sql=library_field_in.db3 \
```

```

        add_field=$add_field"
lc_shell> check_library -logic_library_name \
        "lib1.db lib2.db lib3.db"

```

SQL Syntax for Data Analysis

SQL is used in relational databases. The following tables list the commonly-used SQL syntax required for data analysis.

Table 3-11 Common SQL Keywords

SQL keyword (case-insensitive)	Description
select	Selects data from a database
from	Lists the table to query
group by	Groups the result-set by one or more columns
where	Extracts only the rows that fulfill a specified criterion
as	Assigns an alias to the column
and	Operator. Displays a row if both the first condition AND the second condition are true
or	Operator. Displays a record if either the first condition OR the second condition is true

Table 3-12 Common SQL Wildcard Characters

SQL wildcard character	Description
*	Selects all fields
%	Substitutes one or more characters
-	Substitutes a single character

Table 3-13 Common SQL Functions

SQL Function	Description
round	Rounds a numeric field to the number of decimals specified

Table 3-13 Common SQL Functions (Continued)

SQL Function	Description
count	Returns the number of values of the selected column
min	Returns the smallest value of the selected column
max	Returns the largest value of the selected column
avg	Returns the average value of the selected column
sum	Returns the sum of values of the selected column
abs	Returns the absolute value of the specified numeric expression

Note:

A typical SQL query has the following form:

```
SELECT * FROM table_name WHERE condition
```

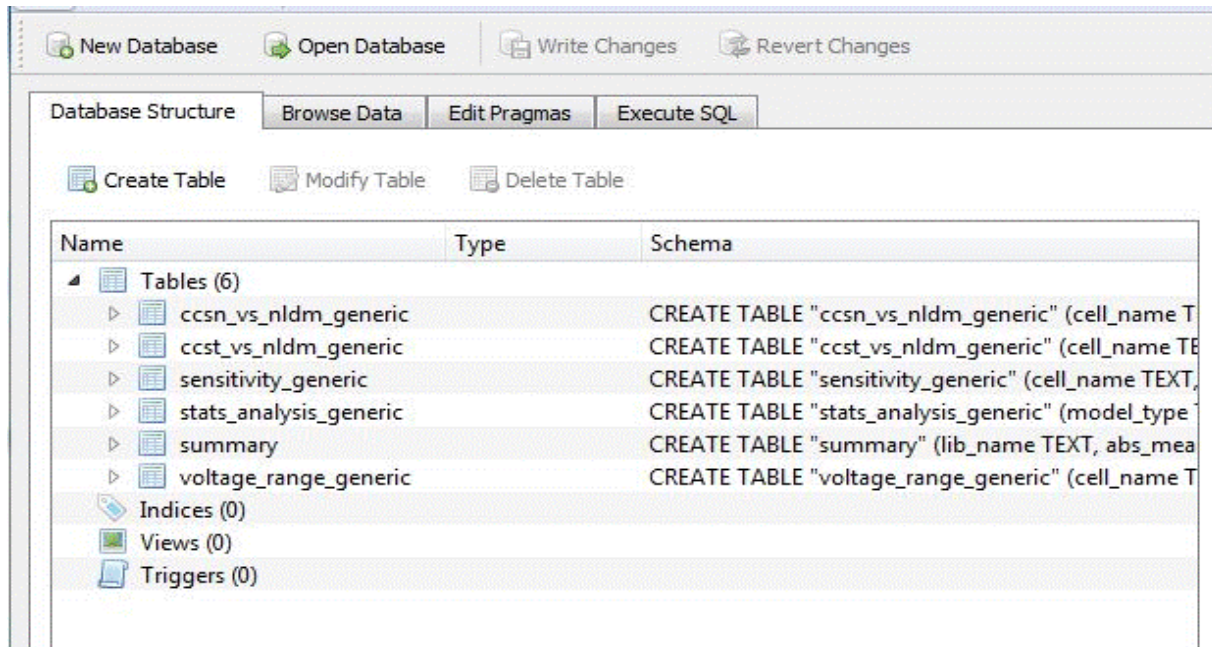
Using the SQLite Browser

The SQLite browser has three main tabs:

- The Database Structure tab
- The Browse Data tab
- The Execute SQL tab

The Database Structure tab lists all the tables contained in the SQL database. For example, the database shown in [Figure 3-8](#) contains six tables, such as `ccsn_vs_nldm_generic` and `ccst_vs_nldm_generic`.

Figure 3-8 Database Structure Tab in SQLite Browser



In the Browse Data tab, you can select a table from the drop-down list. The Browse Data tab displays the data from the selected table in columns. In [Figure 3-9](#), the Browse Data tab

displays the columns, such as `cell_name` and `arc_name`, for the `ccsn_vs_nldm_generic` table.

Figure 3-9 Browse Data Tab in SQLite Browser

	lib_name	abs_mean_error	rel_mean_error	abs_std_dev	rel_std_dev	cell_pass_rate
	Filter	Filter	Filter	Filter	Filter	Filter
1	TT_1p14_25	0.000809783	0.0091	0.00690392	0.0457	0.4
2	TT_1p14_35	0.000802141	0.0092	0.00682759	0.0469	0.4
3	TT_1p155_25	0.00076165	0.009	0.00663566	0.0466	0.4
4	TT_1p155_35	0.00075026	0.009	0.00657694	0.0492	0.4
5	TT_1p17_25	0.000725873	0.009	0.00638675	0.0481	0.4
6	TT_1p17_35	0.000715454	0.0093	0.00635899	0.0674	0.4
7	SS_0p95_125	0.00149916	0.0121	0.0100963	0.0507	0.4
8	SS_0p95_135	0.00141975	0.0117	0.00985557	0.0502	0.4
9	SS_0p99_125	0.00119218	0.0109	0.00884423	0.0488	0.4
10	SS_0p99_135	0.00115659	0.0107	0.00863135	0.0488	0.4
11	SS_1p08_125	0.000807821	0.0094	0.00683828	0.0582	0.4
12	SS_1p08_135	0.000789635	0.0087	0.00675001	0.0718	0.4
13	FF_1p19_m40	0.000645073	0.0083	0.00631541	0.048	0.4

As shown in [Figure 3-10](#), the `summary` table displays the list of summary statistics for each checked library. The column names are `lib_name`, `abs_mean_error`, `rel_mean_error`,

abs_std_dev, rel_std_dev, and cell_pass_rate. Each row shows the summary statistics for a specific library.

Figure 3-10 summary Table in SQLite Browser

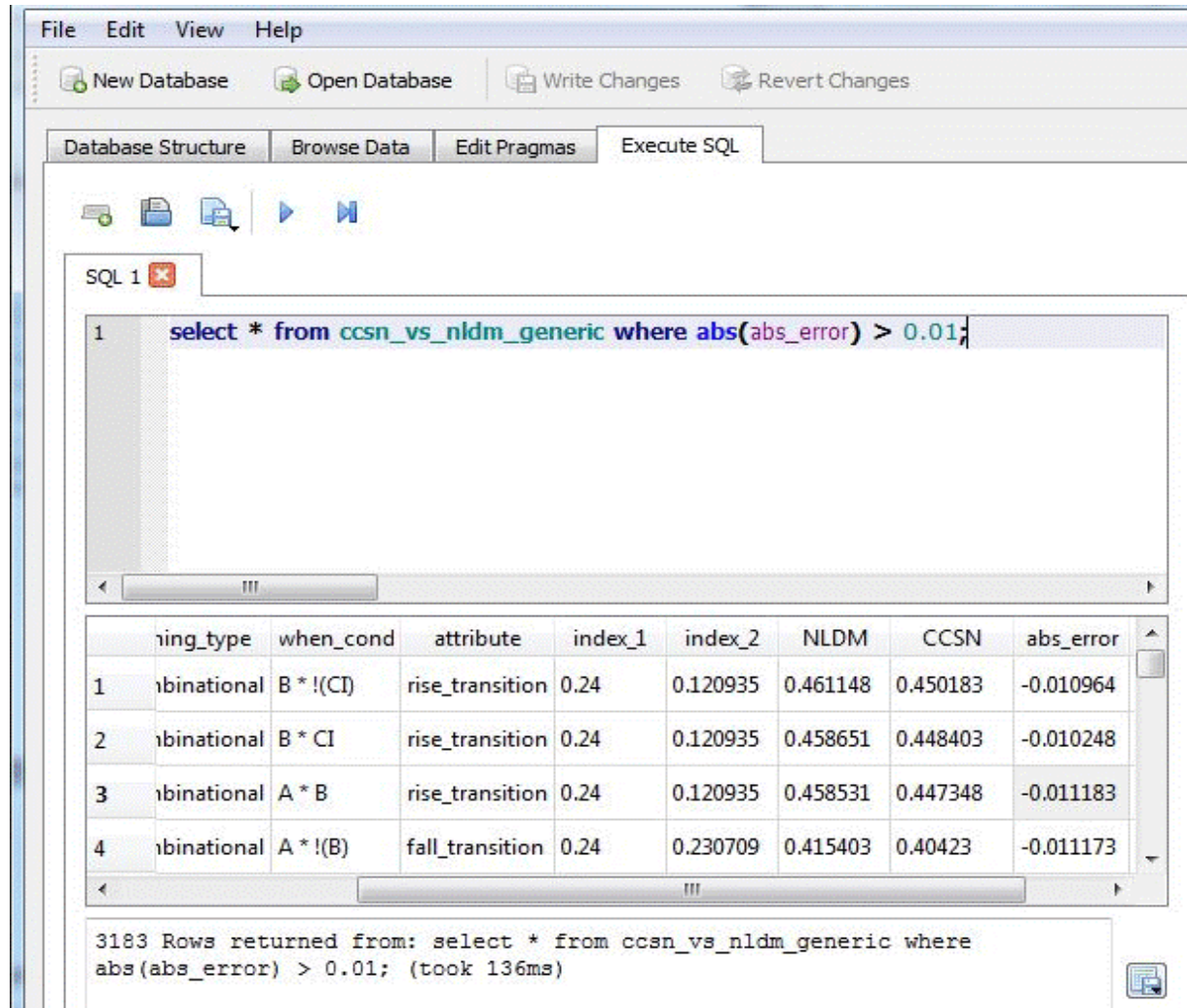
	lib_name	abs_mean_error	rel_mean_error	abs_std_dev	rel_std_dev	cell_pass_rate
	Filter	Filter	Filter	Filter	Filter	Filter
1	TT_1p14_25	0.000809783	0.0091	0.00690392	0.0457	0.4
2	TT_1p14_35	0.000802141	0.0092	0.00682759	0.0469	0.4
3	TT_1p155_25	0.00076165	0.009	0.00663566	0.0466	0.4
4	TT_1p155_35	0.00075026	0.009	0.00657694	0.0492	0.4
5	TT_1p17_25	0.000725873	0.009	0.00638675	0.0481	0.4
6	TT_1p17_35	0.000715454	0.0093	0.00635899	0.0674	0.4
7	SS_0p95_125	0.00149916	0.0121	0.0100963	0.0507	0.4
8	SS_0p95_135	0.00141975	0.0117	0.00985557	0.0502	0.4
9	SS_0p99_125	0.00119218	0.0109	0.00884423	0.0488	0.4
10	SS_0p99_135	0.00115659	0.0107	0.00863135	0.0488	0.4
11	SS_1p08_125	0.000807821	0.0094	0.00683828	0.0582	0.4
12	SS_1p08_135	0.000789635	0.0087	0.00675001	0.0718	0.4
13	FF_1p19_m40	0.000645073	0.0083	0.00631541	0.048	0.4

Use the Execute SQL tab to run SQL queries. The queries can have one or more SQL statements separated by semicolons. In [Figure 3-11](#), the query is:

```
select * from ccsn_vs_nldm_generic where abs(abs_error)> 0.01 ;
```

The query result is a table with rows that meet the search criteria.

Figure 3-11 Execute SQL Tab in SQLite Browser



Analyzing Library Data Using SQL Queries in SQLite Browser

This section shows how to use the SQL queries in the SQLite browser for library data analysis. In all the examples, the library name is `generic`.

The following example shows how to select the rows from the CCS noise versus NLDM validation report with absolute error greater than 0.01.

```
select * from ccsn_vs_nldm_generic where abs(abs_error) > 0.01;
```

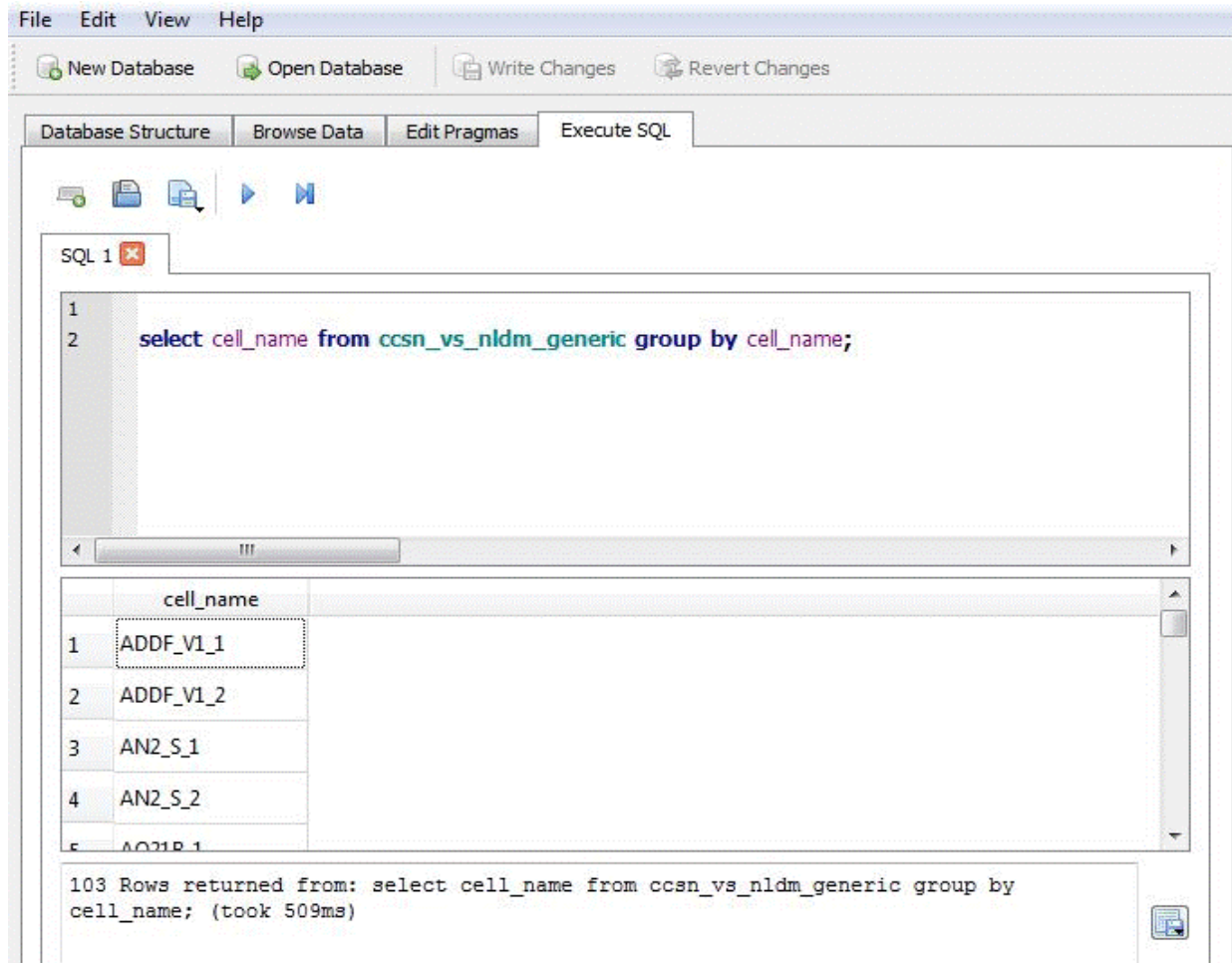
Figure 3-11 shows the query and the results in the Execute SQL tab of the SQLite browser.

The following example shows how to obtain the number and names of cells that fail the `check_library` run. To select the rows from the CCS noise versus NLDM validation report that contain the failing cell count and cell names, use the query:

```
select cell_name from ccsn_vs_nldm_generic group by cell_name;
```

Figure 3-12 shows 103 cell names that match the query.

Figure 3-12 Querying for Cells That Fail `check_library` Run



The following example shows how to plot a histogram using the absolute error data where each bin has a fixed width of 0.01.

```
select round(abs_error/0.01,0)*0.01 as bin, count(*) from
ccsn_vs_nldm_generic group by 1;
```

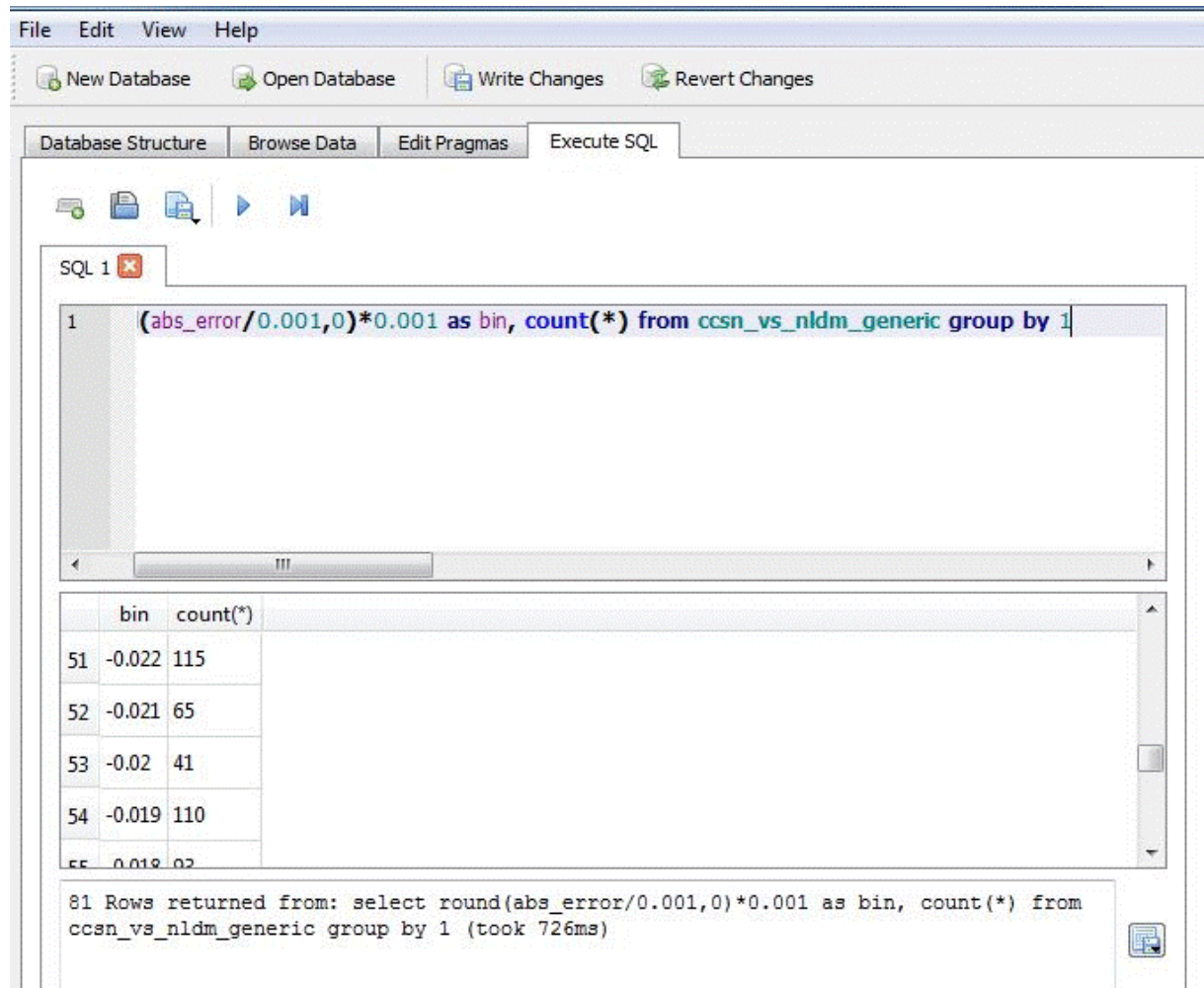
If `$bin_width` is the data width of `abs_err`, the query takes the following form:

```
select round(abs_error/$bin_width,0)*$bin_width as bin, count(*) from
ccsn_vs_nldm_test group by 1;
```

For example, if the `abs_error` data ranges from -0.01 and 0.02 and you want to have 30 bins, you can set the `bin_width` as $(\text{abs}(-0.01) + 0.02)/30 = 0.03/30 = 0.001$.

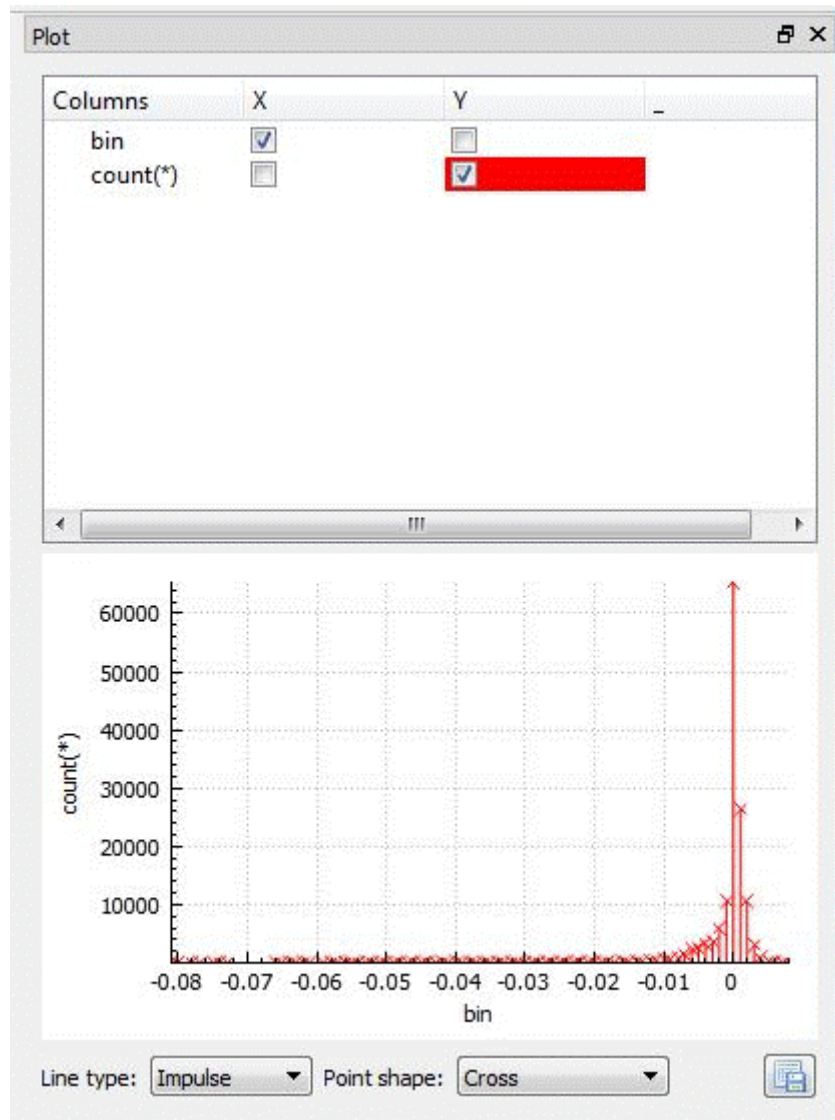
Figure 3-13 shows 81 such bins.

Figure 3-13 Query to Plot a Histogram Using Absolute Error Data With Fixed Bin Width



To generate the plot, check the X and Y fields in the `bin` and the `count` rows as shown.

Figure 3-14 SQLite Browser Plot

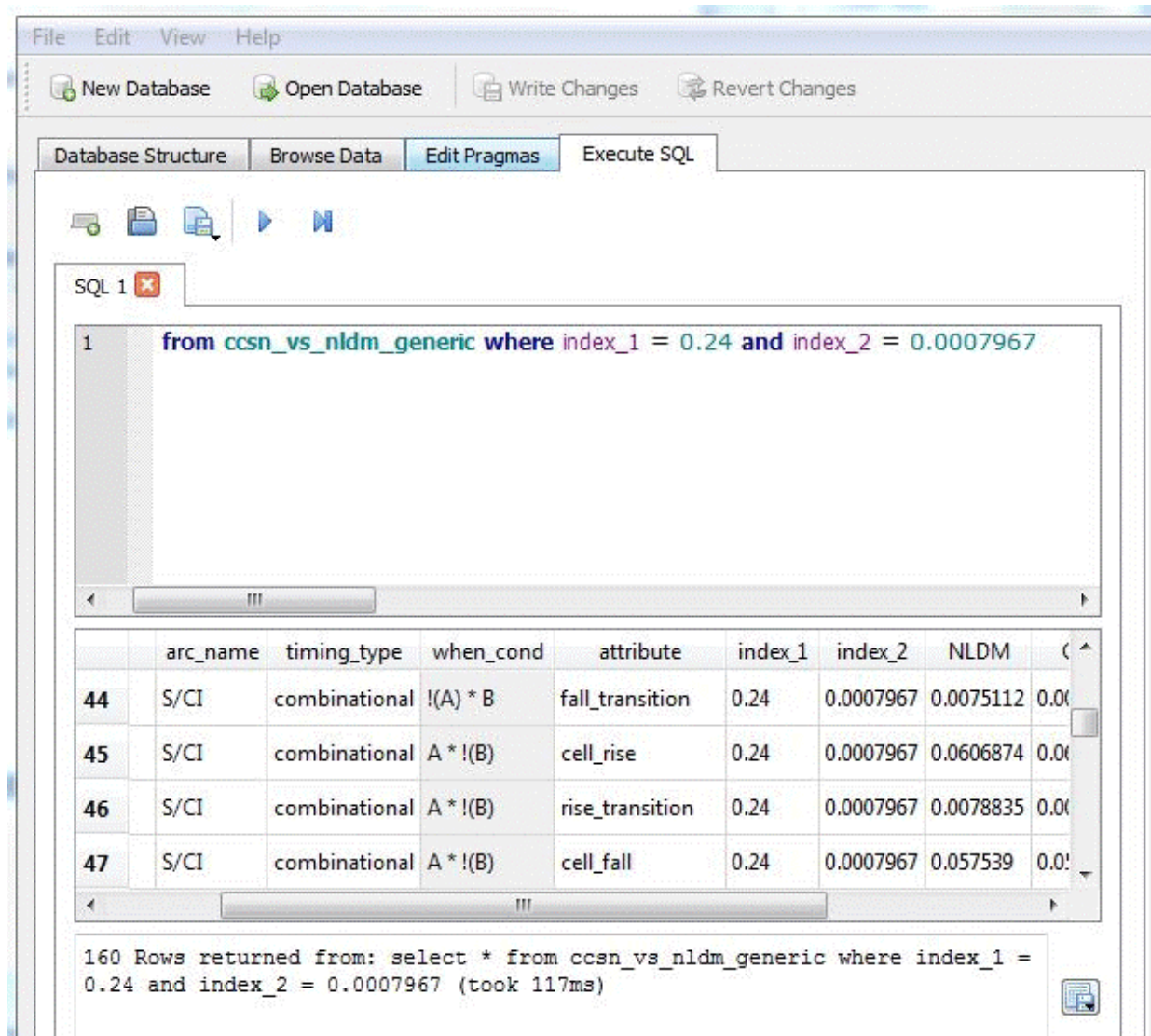


The following example shows how to narrow the search to the specified `index_1` and `index_2` values. Use the `where` keyword to reduce the search space. To obtain the rows with `index_1` value of 0.24 and `index_2` value of 0.0007967, use the query:

```
select * from ccsn_vs_nldm_generic where index_1 = 0.24 and index_2 = 0.0007967;
```

Figure 3-15 shows 160 rows that match the search criteria.

Figure 3-15 Query to Obtain Rows With Specific Attribute Values

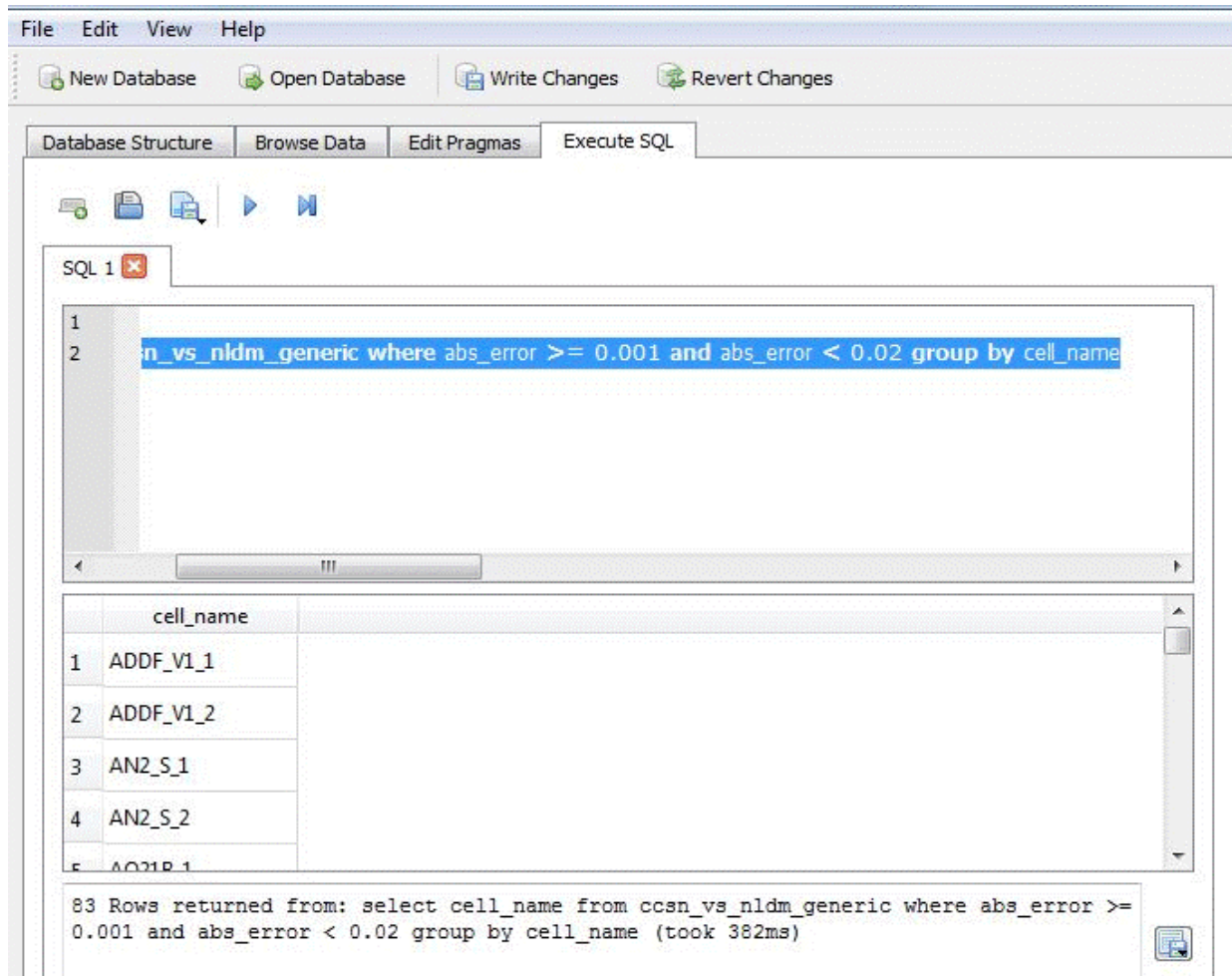


The following example shows how to select the rows where the absolute error is in a specified range. To select the rows where the absolute error is greater than 0.0001 and less than 0.02 and only show the cell names, use the query:

```
select cell_name from ccsn_vs_nldm_generic where abs_error >= 0.0001
and abs_error < 0.02 group by cell_name;
```

Figure 3-16 shows 83 rows that meet the search criteria:

Figure 3-16 Query For Absolute Error Within Specified Range

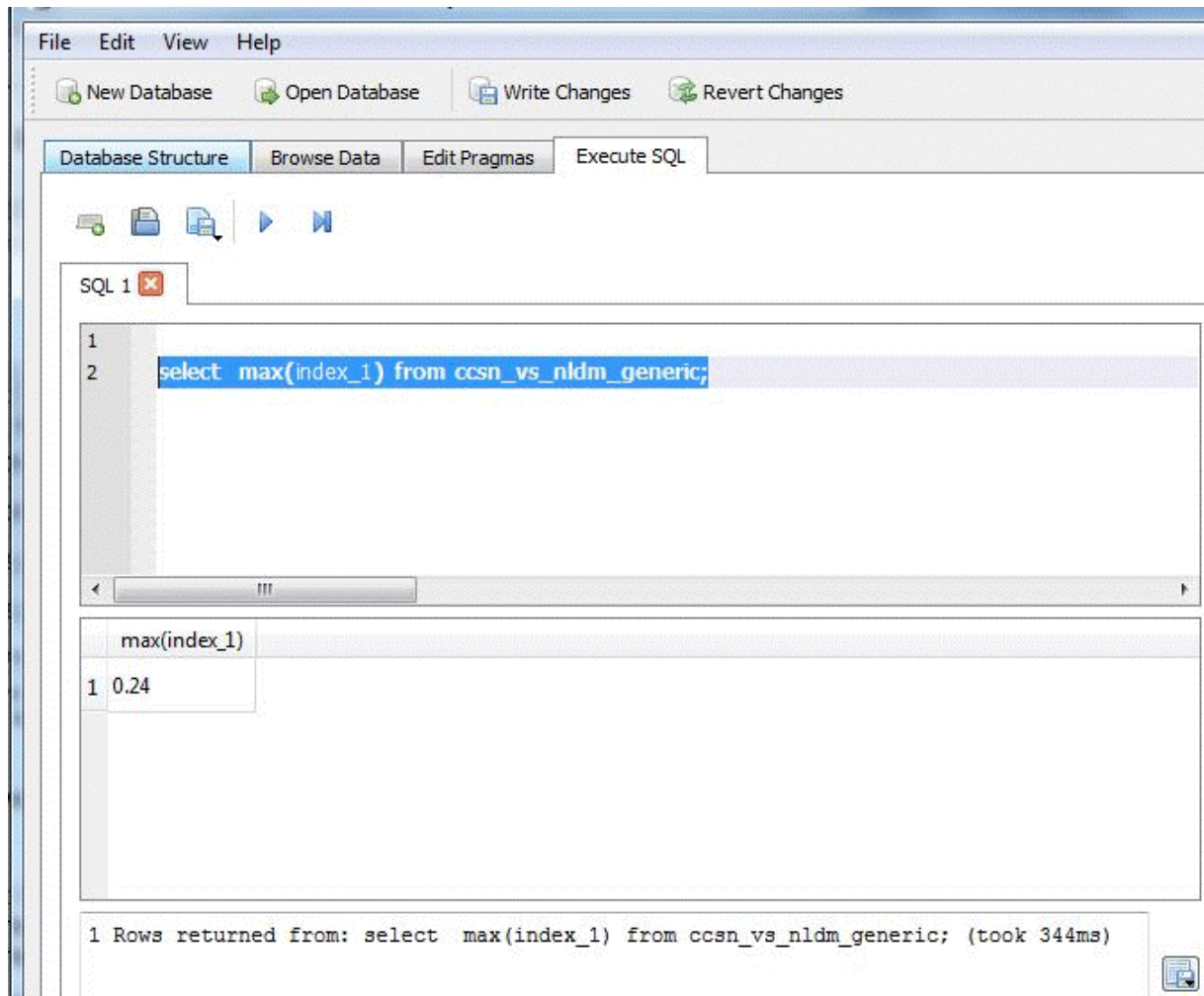


The following example shows how to find the maximum of the `index_1` value. To find the rows with the maximum `index_1` value, use the query:

```
select max(index_1) from ccsn_vs_nldm_generic;
```

Figure 3-17 shows one row with `index_1` value of 0.24.

Figure 3-17 Query for Maximum `index_1` Value



Analyzing Library Data Using SQL-Based Scripts

You can analyze an SQL `check_library` output file using SQL-based scripts. Table 3-14 shows the SQL keywords that are commonly used in these scripts.

Table 3-14 Common SQL Keywords for Use in SQL-Based Scripts

SQL keyword (case-insensitive)	Description
<code>create view</code>	Creates a virtual table view based on the SQL result-set statement

Table 3-14 Common SQL Keywords for Use in SQL-Based Scripts (Continued)

SQL keyword (case-insensitive)	Description
<code>create table</code>	Creates a table in the database
<code>drop table</code>	Removes a table definition and data
<code>eval</code>	Evaluates the SQL statement by taking a character value, interprets it as an SQL statement, and processes it
<code>pragma table_info</code>	Queries information about a specific table
<code>close</code>	Closes the SQL database file

These scripts also use reserved Library Compiler keywords and procedures.

Table 3-15 Reserved SQL Keywords and Procedures for Use in Library Compiler

Library Compiler keywords and procedures	Description
<code>sqlite3 db</code>	Opens the SQL database file
<code>db3 close</code>	Closes the SQL database file
<code>db3 eval</code>	Evaluates the next SQLite statement
<code>write_records_in_csv</code>	Procedure to write SQLite query data to output CSV file

Examples

The following script example creates a table listing the failing cell names (if any) from the `ccsn_vs_nldm` table in the SQL file and outputs them to a CSV file.

The script performs these actions in the following order:

- Checks the existence of the `ccsn_vs_nldm` table in the SQL file, `test.db3`
- Creates an on-the-fly table, `fail_cells`
- Selects the `cell_name` field from the `fail_cells` table
- Writes out the `cell_name` field to the output file, `ccsn_vs_nldm_failing_cells.csv`

```
sqlite3 db3 test.db3
if {[llength [db3 eval {pragma table_info(ccsn_vs_nldm)}] ] > 0} {
```

```

db3 eval {create view if not exists fail_cells as select cell_name
from ccsn_vs_nldm group by cell_name}
set tbl_exist [db3 eval {pragma table_info(fail_cells)}]
if {[length $tbl_exist] > 0} {
    set array [db3 eval {select cell_name from fail_cells}]
}
set out_file ccsn_vs_nldm_failing_cells.csv
set file_id [open $out_file w]
puts $file_id "cell_name"
write_records_in_csv $file_id $array 1
close $file_id
} else {
    echo " No ccsn_vs_nldm table exists \n"
}

```

The following script example creates a histogram table from the `ccsn_vs_nldm` table in the SQL file and outputs them to a CSV file.

The script performs these actions in the following order:

- Checks the existence of the `ccsn_vs_nldm` table in the SQL file, `test.db3`
- Creates an on-the-fly table, `histo`
- Selects all the data from the `histo` table
- Writes out the selected data to the output file, `ccsn_vs_nldm_histogram.csv`

```

sqlite3 db3 test.db3
set bin_width 0.001
if {[length [db3 eval {pragma table_info(ccsn_vs_nldm)}] ] > 0} {
    db3 eval {drop table if exists histo}
    db3 eval {create table histo as select round(abs_error/$bin_width)*
$bin_width as bin, count(*) from ccsn_vs_nldm group by 1}
    set arr3 [db3 eval {select * from histo}]
    set n_out_file ccsn_vs_nldm_histogram.csv
    set n_file_id [open $n_out_file w]
    puts $n_file_id "error_bin,number_of_errors"
    write_records_in_csv $n_file_id $arr3 2
    close $n_file_id
} else {
    echo " No ccsn_vs_nldm table exists \n"
}
db3 close

```

4

Library Quality Assurance Flows

This chapter describes the library quality assurance commands you can use to ensure that the following library models are complete, consistent, and accurate: timing, noise, power, scaling, multicorn-multimode, and IEEE 1801, also known as Unified Power Format (UPF).

This chapter contains the following sections:

- [Flows for Timing Models](#)
- [Flows for CCS Noise Models](#)
- [Flows for Power Models](#)
- [Flows for UPF Models](#)
- [Flows for Scaling Libraries](#)
- [Flows for Multicorner-Multimode Libraries](#)

Flows for Timing Models

The following sections describe how to use the following library quality assurance commands for timing models:

- `read_lib`
Performs syntax checks, semantics checks, and screener checks.
- `check_library`
Checks the quality of individual libraries and the consistency between libraries.
- `report_lib`
Reports timing data.

Using the `read_lib` Command to Check Delay and Constraint Models

The `read_lib` command performs syntax checks, semantics checks, and screener checks. It verifies that the library model is complete, that the library complies with Liberty rules, and that the library is consumable for downstream Synopsys tools. The `read_lib` command also verifies whether the library cell has the constraint models that are required for timing analysis.

To run the `read_lib` command, enter the following at the shell prompt:

```
prompt> read_lib generated_library.lib
```

The `read_lib` command issues a warning or error message if it finds a problem in the library, such as an incorrectly defined value or a missing attribute or table (if the attribute or table is required).

For information about screener checks for delay and constraint models, see [Chapter 1, “Reading and Compiling Libraries.”](#)

Using the `check_library` Command to Check Delay Models

Library delay values can be represented in both NLDM and CCS timing models. The NLDM and CCS timing models in a library should be consistent, meaning that the NLDM and CCS timing values in the library for a cell arc should yield the same timing values for a given index or grid point. You can check the consistency between NLDM and CCS timing models in a library by using the `check_library` command.

The `check_library` command set includes the `set_check_library_options` command, which uses options to perform specific checks. The following examples show the commands you can use to check NLDM and CCS timing models for consistency:

```
prompt> set_check_library_options -validate {timing} \
      -tolerance {delay 0.02 0.002 slew 0.05 0.005} \
      -report_format {csv=output_results nosplit}

prompt> check_library -logic_library_name "../output.db"
```

You can specify both the absolute and relative tolerance to identify the delay and slew differences between NLDM and CCS timing models. Use the `set_check_library_options` command to specify the delay and slew tolerances between NLDM and CCS timing values, as shown:

```
prompt> set_check_library_options -tolerance \
      {type relative_tolerance absolute_tolerance}
```

The validation results are published in the standard output format and the CSV format. The following example shows the `check_library` validation results in standard output format:

```
#BEGIN_XCHECK_VALIDATION
Comparison result saved in output_results/validate_timing directory.
No inconsistent data found for delay values
No inconsistent data found for slew values
#END_XCHECK_VALIDATION
Logic vs. logic library check summary:
Logic library consistency check PASSED for timing validation.
#END_XCHECK_LIBRARY
```

Any mismatch identified between NLDM and CCS timing is published in the CSV format.

For more information about `check_library` command checks for NLDM and CCS timing models, see [Chapter 3, “Library Checking.”](#)

Using the `check_library` Command to Check Constraint Models

The `check_library` command provides checks for constraint models. *Constraint* refers to the setup, hold, recovery, removal, minimum pulse width, nonsequential setup, and nonsequential hold tables in a library.

The calculated constraint value in the library depends on the constraint calculation methodology. For example, to calculate a setup constraint value, you can use the delay degradation or pass-fail methodology. There are other methodologies to calculate a constraint value in a library. To verify the constraint numbers in a library, you must know the constraint calculation methodology. The following sections describe the various constraint methodologies and the quality assurance methodologies available to validate constraint models.

The constraint models in a library generally follow a monotonic trend, though it is not a requirement. For example, the constraint values increase as the input slew and output load increase. In some cases, a cell behaves nonlinearly, disrupting this monotonic behavior. Because most constraint models follow a monotonic trend, an exceptionally large outlier in the middle of the table is worth investigating. To see a table that describes the possible table value trends, see the “Trend Analysis for Single Characterization Tables” section in [Chapter 3, “Library Checking.”](#)

The `check_library` command allows you to check the constraint model trends in a library. Trend analysis reports the table value variation along a row or column of a constraint lookup table (LUT) with respect to input slew and output load capacitance change.

The following example shows the commands you can use to perform `check_library` trend analysis:

```
prompt> set_check_library_options -group_attribute \
        {*constraint} -analyze {table_trend} -criteria \
        {trend=^ trend=V trend=M trend=W trend=N } \
        -tolerance {constraint 0.3 0.100}

prompt> check_library -logic_library_name "output.db"
```

The specified tolerances are

```
-tolerance {type relative_tolerance absolute_tolerance}
```

The previous commands perform trend analysis on the `output.db` library for all constraint models. The `check_library` command compares the adjacent constraint values in a table and highlights those lines in the table that violate the specified tolerances. Also, it publishes the trends that the violated values follow.

For more information about `check_library` command checks for constraint models, see [Chapter 3, “Library Checking.”](#)

Using the `report_lib` Command to Generate Timing Reports

Use the `report_lib` command to generate reports that contain library information. If you do not specify any command options, the `report_lib` command displays a default report containing library-level data and information about the available cells.

You can use the following options with the `report_lib` command to generate timing reports for generic CMOS and CMOS nonlinear delay models:

```
-timing
```

Displays pin-level timing information for constraints or delays.

Note:

The `-timing` option is available only after you read the library source file (.lib) into the tool's memory.

-timing_arcs

Displays all the timing arc information in the library. When you use this option, the report lists the arc sense, type, from and to pins, and the `when` statement of all cells.

-timing_label

Displays all the timing arc label information.

For more information and specific examples, see [Chapter 2, "Generating Library Reports."](#)

Flows for CCS Noise Models

CCS noise characterization data provides information about noise failure detection on cell inputs, calculation of noise bumps on cell outputs, and noise propagation through the cell. For best accuracy, you must add CCS timing data to the library in addition to the CCS noise data. CCS noise data includes the following:

- Channel-connected block parameters
- DC current tables
- Timing tables for rising and falling transitions
- Timing tables for low and high propagated noise

The following sections describe how to use the following library quality assurance commands for CCS noise models:

- `read_lib`
Performs syntax checks, semantics checks, and screener checks.
- `report_lib`
Reports CCS noise data.

Using the `read_lib` Command to Check CCS Noise Data

Use the `read_lib` command in the Library Compiler tool to check the CCS noise data in your library. The `read_lib` command performs syntax checks, semantics checks, and screener checks. It verifies that the library model is complete, that the library complies with Liberty rules, and that the library is consumable for downstream Synopsys tools.

To run the `read_lib` command, enter the following at the shell prompt:

```
prompt> read_lib generated_library.lib
```

The `read_lib` command issues a warning or error message if it finds a problem in the library, such as an incorrectly defined value or a missing attribute or table (if the attribute or table is required).

For information about screener checks for CCS noise models, see [Chapter 1, “Reading and Compiling Libraries.”](#)

Using the `report_lib` Command to Report CCS Noise Data

Use the `report_lib` command to generate reports that contain library information. If you do not specify any command options, the `report_lib` command displays a default report containing library-level data and information about the available cells.

You can specify the `-noise_arcs` option with the `report_lib` command to display CCS noise conditional data modeling information with the `mode` or `when` attributes. When you specify the `-noise_arcs` option, `report_lib` displays the CCS-noise-related arc and pin information using the `ccsnf` (CCS noise first stage) and `ccsnl` (CCS noise last stage) attributes.

For more information and specific examples, see [Chapter 2, “Generating Library Reports.”](#)

Flows for Power Models

The library NLPM format captures leakage power numbers in multiple input combinations to generate a state-dependent table. It also captures the dynamic power of various input transition times and output load capacitance to create the state-dependent and path-dependent internal energy data.

NLPM models consist of `leakage_power` and `internal_power` library tables. The calculated power values in a library depend on the tool’s power calculation methodology. For example, the calculated leakage power in a library might or might not include gate leakage power. Depending on user settings, this gate leakage power might be included in the `internal_power` table instead of the `leakage_power` table. When publishing the propagating internal power in a library, the corresponding nonpropagating internal power should be subtracted to avoid any double counting in the power numbers. There are other criteria for calculating NLPM values in a library. To verify the NLPM values in a library, you should know the characterization tool’s power calculation methodology.

The CCS power modeling format extends current library models to include current-based waveform data to provide a complete solution that addresses static and dynamic power. It

also addresses dynamic IR drop. CCS power models create a single unified power library format suitable for power optimization, power analysis, and rail analysis.

The following sections describe how to use the following library quality assurance commands for power models:

- `read_lib`
Performs syntax checks, semantics checks, and screener checks.
- `report_lib`
Reports power data.

Using the `read_lib` Command to Check NLPM and CCS Power Data

Use the `read_lib` command in the Library Compiler tool to check the NLPM and CCS power data in your library. The `read_lib` command performs syntax checks, semantics checks, and screener checks. It verifies that the library model is complete, that the library complies with Liberty rules, and that the library is consumable for downstream Synopsys tools.

The `read_lib` command verifies whether an NLPM model in a library meets Liberty standards. It also checks the following CCS power model syntax: leakage current, gate leakage, intrinsic parasitic model, dynamic power model, power and ground current, compact CCS power, and variation-aware CCS power leakage current.

To run the `read_lib` command, enter the following at the shell prompt:

```
prompt> read_lib generated_library.lib
```

The `read_lib` command issues a warning or error message if it finds a problem in the library, such as an incorrectly defined value or a missing attribute or table (if the attribute or table is required).

For information about screener checks for NLPM and CCS power models, see [Chapter 1, “Reading and Compiling Libraries.”](#)

Using the `report_lib` Command to Report Power Data

Use the `report_lib` command to generate reports that contain library information. If you do not specify any command options, the `report_lib` command displays a default report containing library-level data and information about the available cells.

You can specify the following options with the `report_lib` command to generate power reports:

`-power`

Displays power-related information, including internal power, leakage power, and multiple power supplies.

`-power_label`

Displays all power label information.

For more information and specific examples, see [Chapter 2, “Generating Library Reports.”](#)

Flows for UPF Models

The following sections describe how to use the following library quality assurance commands for UPF models:

- `read_lib`
Performs syntax checks, semantics checks, and screener checks.
- `check_library`
Checks the quality of individual libraries and the consistency between libraries.
- `report_lib`
Reports PG pin data.

Using the `read_lib` Command to Check UPF Data

Use the `read_lib` command in the Library Compiler tool to check the UPF data in your library. The `read_lib` command performs syntax checks, semantics checks, and screener checks. It verifies that the library model is complete, that the library complies with Liberty rules, and that the library is consumable for downstream Synopsys tools.

Use the `read_lib` command, as shown, to confirm that no critical warning or error messages were generated for the PG pin library:

```
prompt> read_lib pg.lib
```

If the library did not compile successfully, check the specific error and fix the problem as necessary. The Library Compiler tool can fail to compile the library in the following circumstances:

- A Library Compiler check introduced in the PG pin syntax

- User bypass of all command checks, which adds unknown Liberty syntax that is not supported in the Library Compiler tool

For information about screener checks for PG pin library models, see [Chapter 1, “Reading and Compiling Libraries.”](#)

Using the `check_library` Command to Check UPF Data

You can check the consistency of UPF library models using the `check_library` command. The `check_library` command set includes the `set_check_library_options` command, which uses options to perform specific checks. To check UPF libraries, specify the `set_check_library_options` command with the `-upf` option. When you do this, the command performs UPF checks for level-shifter cells, isolation cells, retention cells, switch cells, always-on cells, PG pin and partial PG pin cells, and checks for substrate bias cells.

For more information about `check_library` command checks for PG pin library models, see [Chapter 3, “Library Checking.”](#)

Using the `report_lib` Command to Report PG Pin Data

Use the `report_lib` command to generate reports that contain library information. If you do not specify any command options, the `report_lib` command displays a default report containing library-level data and information about the available cells.

You can use the `-pg_pin` option with the `report_lib` command, as shown, to report all PG pin attributes:

```
prompt> report_lib -pg_pin library_name
```

For more information and specific examples, see the “Power and Ground (PG) Pin Information” section in [Chapter 2, “Generating Library Reports.”](#)

Flows for Scaling Libraries

You can determine whether a library is unsuitable for scaling by using the `check_library` command. The `check_library` command set includes the `set_check_library_options` command, which uses options to perform specific checks. Use the following options with the `set_check_library_options` command to perform checks for scaling:

`-scaling timing`

Performs checks for CCS timing scaling.

`-scaling noise`

Performs checks for CCS noise scaling.

`-scaling power`

Performs checks for CCS or NLPM power scaling.

For more information about `check_library` command checks for scaling, see [Chapter 3, “Library Checking.”](#)

Flows for Multicorner-Multimode Libraries

The `check_library` command provides logic library consistency checking for multicorner-multimode flows. The `check_library` command set includes the `set_check_library_options` command, which uses options to performs specific checks. Use the `-mcm` option with the `set_check_library_options` command to specify checks for a multicorner-multimode flows.

For more information about `check_library` command checks for multicorner-multimode flows, see [Chapter 3, “Library Checking.”](#)