Qualib User Guide

Product Version 2015.12

Huada Empyrean Software Co., Ltd.
www.empyrean.com.cn

Copyright and Proprietary Information

Trademarks: Qualib is the trademark of Huada Empyrean Software Co., Ltd. All other trademarks are the property of their respective holders. For queries regarding Empyrean's trademarks, contact our company at the address shown below or call +8610-84776888.

Contact Information
Huada Empyrean Software Co., Ltd.
2/F Building A，NO.2，Lizezhonger Road，Chaoyang District，
100102，P.R.China
Email: support@empyrean.com.cn
Telephone: +8610-84776888
http://www.empyrean.com.cn

Preface

About This Guide

Qualib provides a platform to check the correctness and consistency of all kinds of library files, and also the functionalities for debugging, cross-reference and report generation. This guide describes the major features of Qualib.

Organization

This manual contains the following chapters:

- Chapter 1, Introduction, provides the overview introduction of Qualib.
- Chapter 2, Usage Flow, describes the usage flow.
- Chapter 3, Inspection Rules, describes the checking rule parameters.
- Chapter 4, Checking, describes the checking of library files.
- Chapter 5, Cross-Checking, describes the cross-checking of library files.
- Chapter 6, Self-Checking, describes the self-checking of library files.
- Chapter 7, Comparison, describes the comparison of library files.
- Chapter 8, Setting Preferences, describes the preference settings.
- Chapter 9, Library View Windows, introduce the library viewer GUIs.

Conventions

The following conventions are used in this document.

Convention	Description
`Courier`	Command syntax `link_gds`
`Courier italic`	Command option `link_gds -session sessionName`
Arial bold	User inputted command qualib > **help**
[]	Optional parameters
|	Choice of alternatives
\	Line continuation
Design → Save	Command sequence, first open Design menu and then select Save

1 Introduction

As the technology node comes to 28nm/20nm and below, designer always have a growing number of complicated library files. It gets harder and harder to check the consistency among those library files, also the correctness between the netlist, layout geometry data, and timing libraries.

Qualib platform aims at providing a system, which can check the integrity and quality of all kinds of library files and compare the difference between two versions of libraries, and also have functionalities for debugging, cross-reference and report generation.

Currently, following format of library files are supported to check:

- LEF
- GDS
- Timing Lib
- Verilog
- CDL
- ATPG
- MBIST
- LVLIB

The checking functionalities include check, cross-check, self-check and comparison:

- Check

  Qualib will do the correctness check for each view library files according to design rules. E.g. for LEF view, the tech definition, pin access, off grid, and off boundary design rule checks will be performed.

- Cross-check

  Qualib will do the cross-check for different view library files. E.g. for given standard cell, it will check the pin definitions in both LEF library and GDS library including pin name, pin layer, pin shapes…

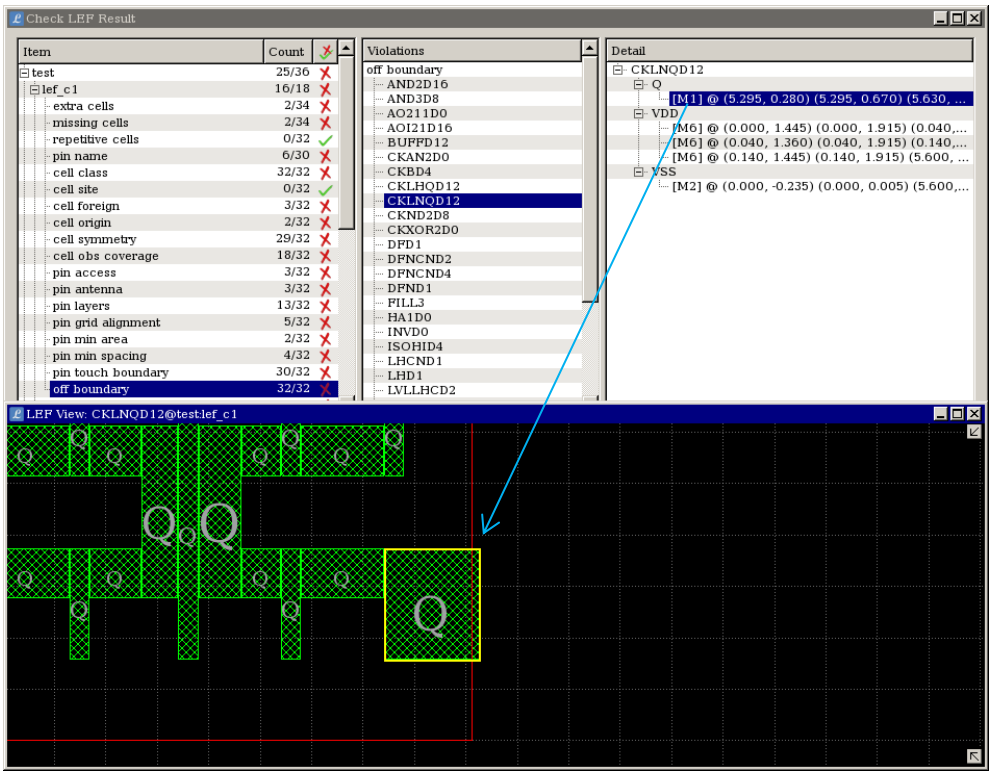
- Self-check

  Qualib will do the self-check for same view library files in different corners. E.g. for timing view library files in fast and slow corner, it will check if the cells are matched and cells have same cell area, pin name, pin direction…

- Comparison

  Qualib will do the comparison for same view library files of different version. E.g. for given timing view library files of two versions, it will compare and analyze the difference of cell area, timing, internal power definitions.

The checking results will be presented in both detail reports and in the GUI window, in which the issues or differences will be highlighted.



2
Usage Flow

Qualib provides a flexible working environment with both a shell command-line interface and a graphical user interface.

This section introduces the flow of using Qualib for library files checking and comparison.

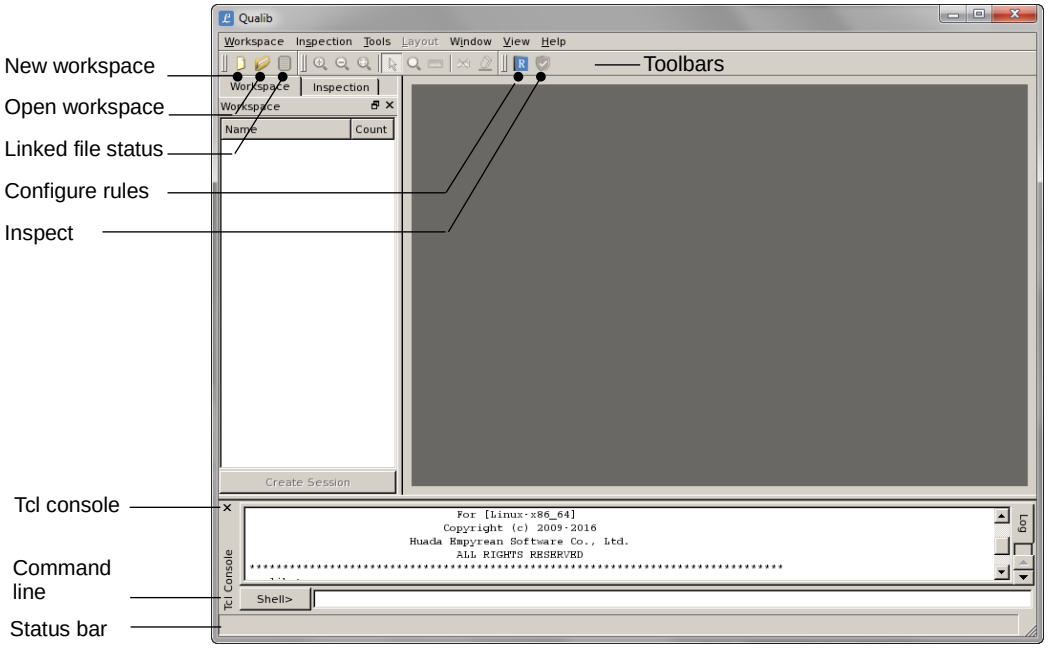
Get Started

Qualib operates in the X Windows environment on Linux. Before starting, make sure the environment variable *$QUALIB_HOME* has been set to the installation directory and also the environment variable *$DISPLAY* is set correctly.

Qualib can be started by entering the *qualib* command in a Linux shell. By default, this command starts the tool in the command-line interface. You can start the tool in the GUI by specifying the *-gui* option.

```
% $QUALIB_HOME/bin/qualib  [-gui]
```

The Qualib main window is shown below:

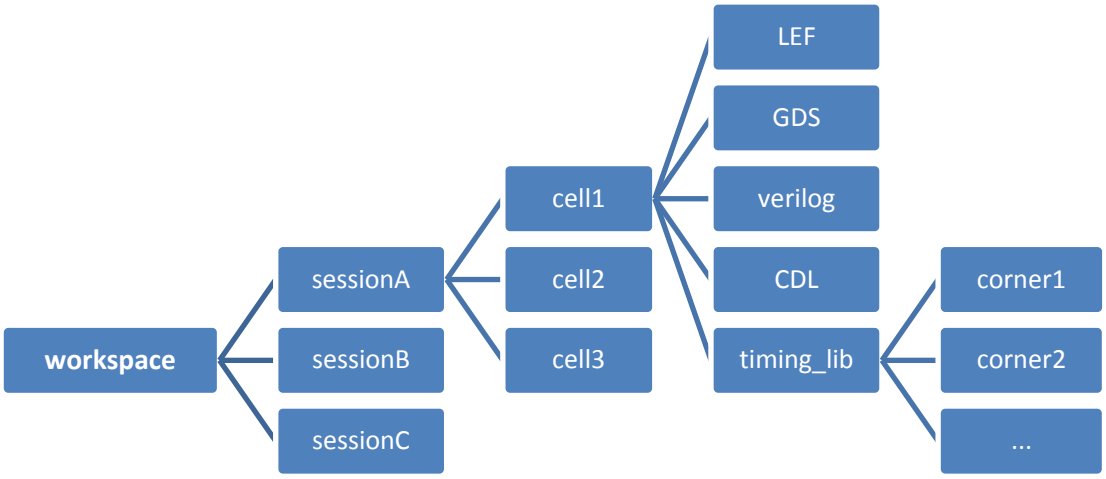


When you start the command-line interface, the qualib > prompt appears in the Linux shell. If you need to use the GUI after starting the command-line interface, enter the start_gui command at the qualib > prompt.

```
qualib > start_gui
```

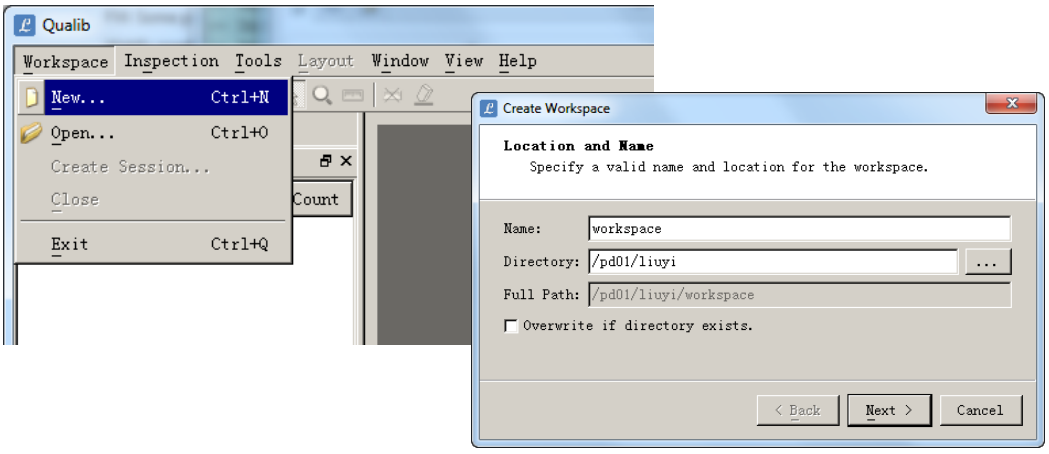
Build Workspace

The library view files are organized as below:



Workspace is the root directory, including all the libraries. One session records one specific version of the library. Under each session, it lists all the cells in the library. For each cell, it may have LEF/GDS/Timing lib/Verilog/CDL/ATPG/MBIST/LVLIB library views. There could be multi corners information under each library view.

Workspace and sessions can be created either from the menu ***Workspace → New …*** or in command shell.

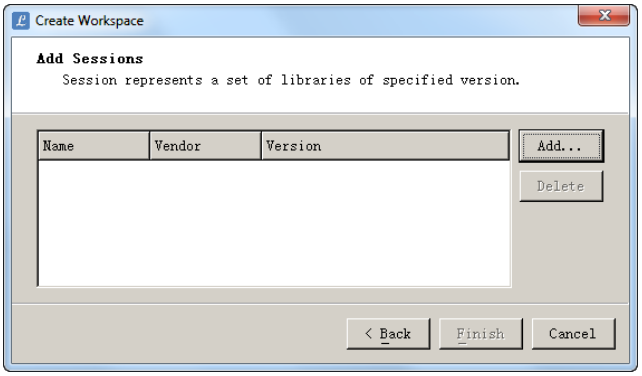


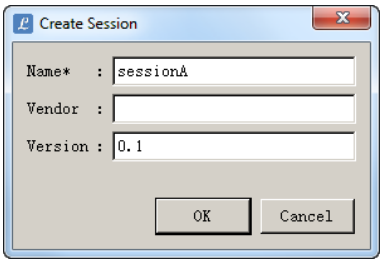

```
qualib > create_workspace    directoryName    [-overwrite]
```

Once the workspace has been created, it can be loaded in the next time.
Also if previous inspection result exists, it will be restored when opening workspace.

qualib > **open_workspace** directoryName

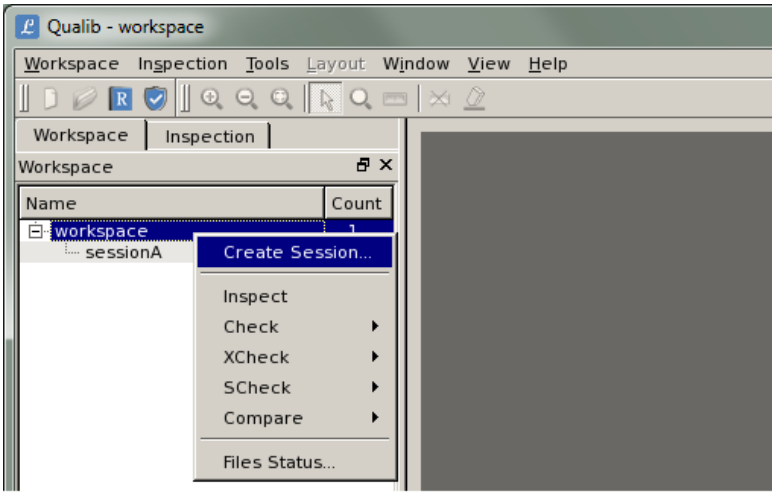
The workspace configuration window will pop up automatically, allowing user to add sessions to the workspace. Each session includes a set of library files.





qualib > **create_session** sessionName [-*version* ver]

You can also add session to workspace from the GUI window.
Right click the workspace name, and select "***Create Session…***" to create new session.

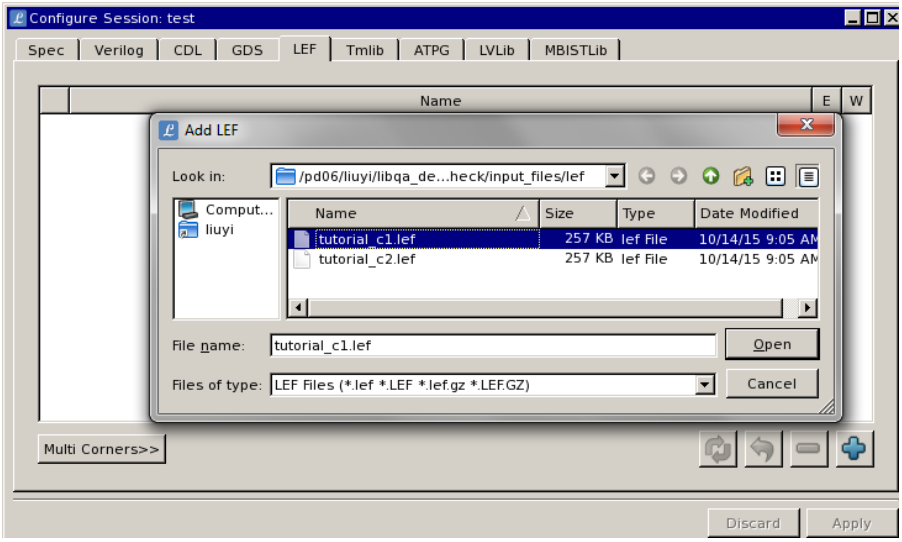


Read Library Files

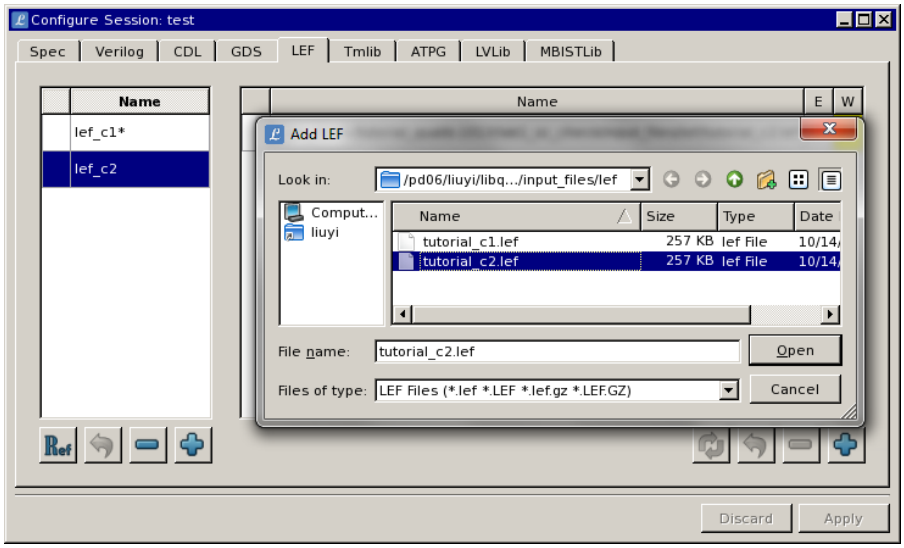
Next step is to read in the related LEF/GDS/Timing lib/Verilog/CDL/… library files, and link them into the corresponding session.

Double click the session name or right click the session and select "***Configure…***" to invoke the session configuration window.

If there is only one corner for some library view, just click the  button to add the corresponding files.



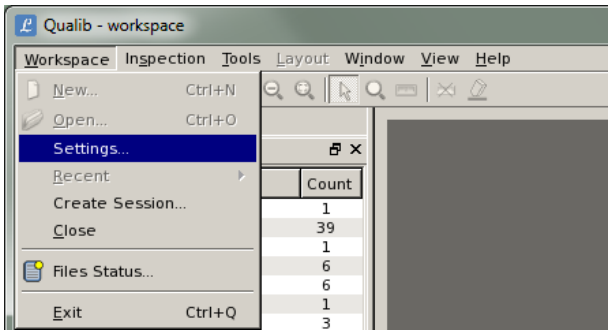
If there are multiple corners for some library view, click the "***Multi Corners***>>" button, and create the corners first, and then click the  button to add the corresponding files. Please note that there will be a "*****" mark in the end of reference corner.

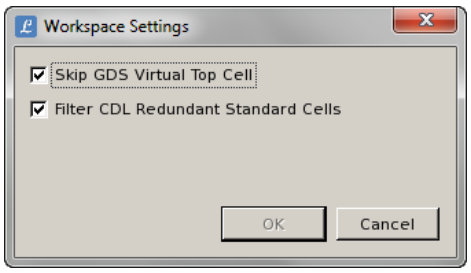


Click the "***Apply***" button, after all the files are specified.

```
qualib > link_spec                  -session  sName  cellListFile
qualib > create_verilolg_corner     -session  sName  cName
qualib > link_verilog               -session  sName  [-corner  cName]  *.v
qualib > create_cdl_corner          -session  sName  cName
qualib > link_cdl                   -session  sName  [-corner  cName]  *.cdl
qualib > create_gds_corner          -session  sName  cName
qualib > link_gds                   -session  sName  [-corner  cName]  -tech  techFile
qualib > link_gds                   -session  sName  [-corner  cName]  *.gds
qualib > create_lef_corner          -session  sName  cName
qualib > link_lef                   -session  sName  [-corner  cName]  *.lef
qualib > create_tmlib_corner        -session  sName  cName
qualib > link_tmlib                 -session  sName  [-corner  cName]  *.lib
qualib > create_atpg_corner         -session  sName  cName
qualib > link_atpg                  -session  sName  [-corner  cName]  *.atpg
qualib > create_lvlib_corner        -session  sName  cName
qualib > link_lvlib                 -session  sName  [-corner  cName]  *.lv
qualib > create_mbistlib_corner     -session  sName  cName
qualib > link_mbistlib              -session  sName  [-corner  cName]  *.mbist
```

In the menu ***Workspace → Settings …*** , it also provides switches when reading in GDS or CDL files.





If "*Skip GDS Virtual Top Cell*" is not selected, or run the command

```
qualib > show_virtual_top_cell    true
```

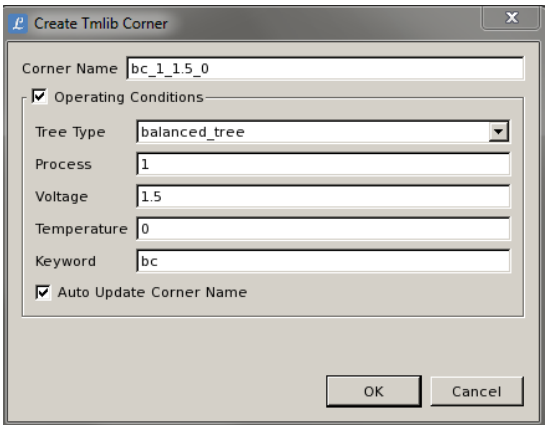
It will show the virtual top cell (having instances but no shape at all) in GDS file. Qualib will skip virtual top cell and trace one level down to get the real cells in file by default. If it is set to show the virtual top cell, virtual cell is treated as normal top cell.

If "*Filter CDL Redundant Standard Cells*" is not selected, or run the command

```
qualib > show_redundant_std_cells    true
```

It will show the redundant standard cells for CDL. For IP, memory or other macro cells, it needs to include the file contains the complete set of standard cells. By default, these standard cells are filtered even if they are not referenced by any cells. If it is set to show the redundant cells, Qualib will fetch all the non-referenced standard cells.

Please note that when creating tmlib corner, PVT (Process/Voltage/Temperature), tree type and the keyword information can be specified.

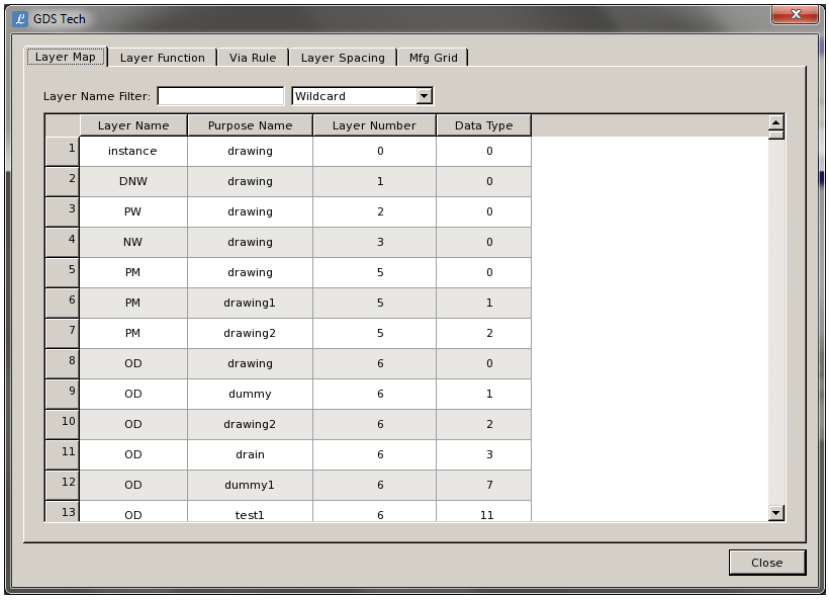



```
qualib > create_tmlib_corner  -session  sName    cName
         -process              1
         -voltage              1.5
         -temperature          0
         -keyword              bc
         -tree_type            balanced_tree
```

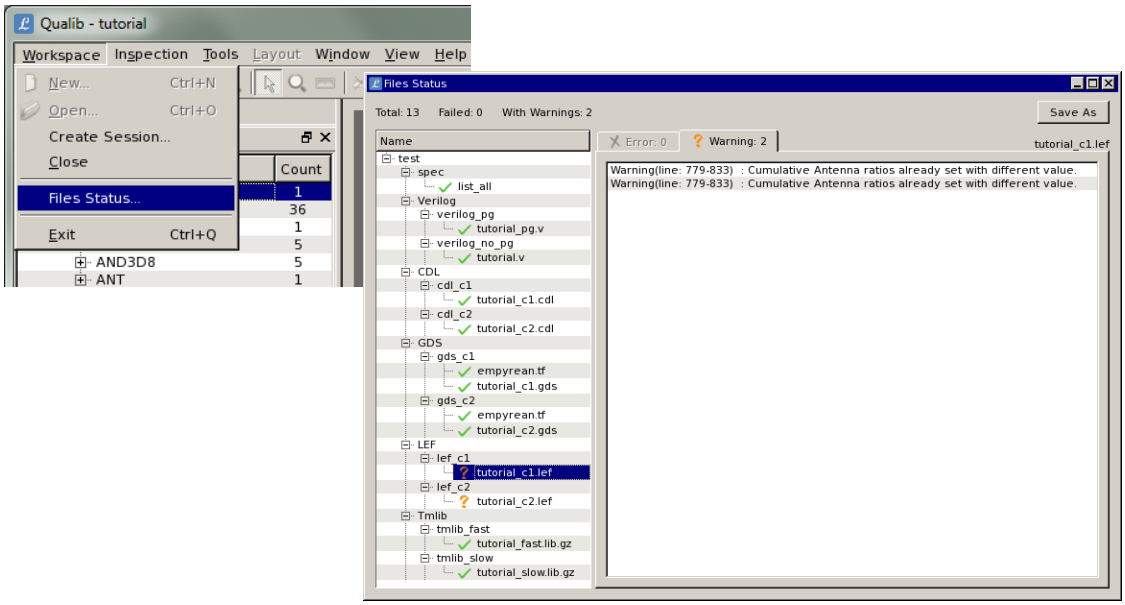
Please note that when linking GDS files, a complete tech file should be provided, because it will be used by most of GDS checking items.

The tech file can be specified to virtuoso, laker tech file or the layer map file, or tech LEF file.

Click the  button to view the contents (layer map/function/spacing/…) in tech file.



User can check the loaded files status from GUI menu ***Workspace → File Status…***
Or right click the workspace name and select "***File Status…***" from the pop up menu.



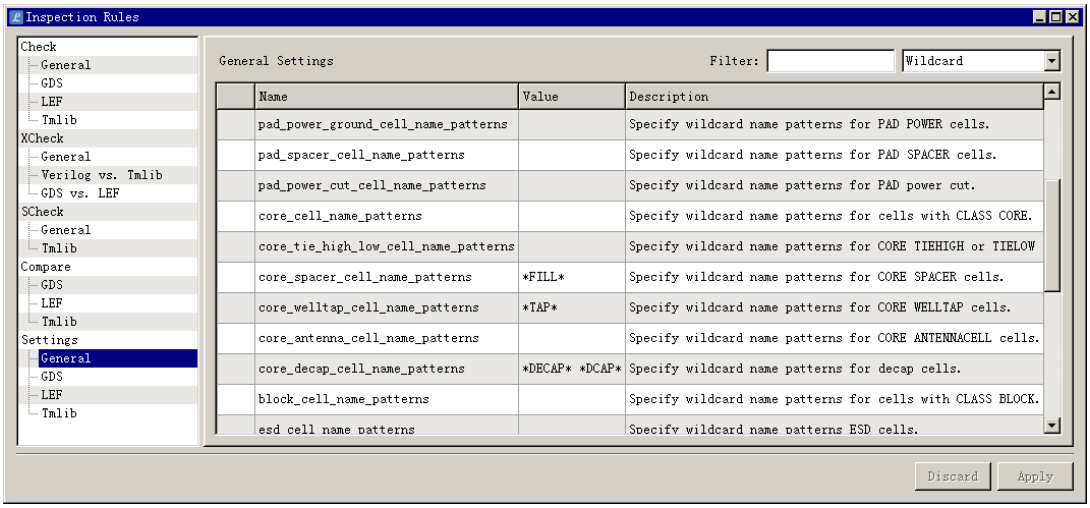
qualib > **report_file_status** [*-verbose*]

If there is any warning or failure during loading the library files, messages will be listed in the "File Status" window. Double click the message, text file viewer will be opened and locate to such line automatically.

Configure Rule Settings

Before the library checking, there are some inspection rule settings to be configured. Select the "***Inspection → Configure Rules…***" menu, or just click the button. The inspection rules window will pop up to configure.

Select the "***Inspection → Load Rules…***" menu, parameter settings can be set to pre-defined parameter settings (for *standard cell*, *IP*, and *IO*), or loaded from file.



For different library type, e.g. *standard cell*, *IP*, and *IO*, user may need to configure different set of "***General Settings***" according to the real library design.

Library Type	Name	Suggested Value
STD CELL	core_cell_name_patterns	*
	core_spacer_cell_name_patterns	*FILL*
	core_tie_high_low_cell_name_patterns	
	core_antenna_cell_name_patterns	
	core_decap_cell_name_patterns	*DECAP* *DCAP*
	core_welltap_cell_name_patterns	*TAP*
	endcap_pre_post_cell_name_patterns	
IO PAD CELL	pad_cell_name_patterns	
	pad_spacer_cell_name_patterns	
	pad_bonding_cell_name_patterns	
	pad_power_ground_cell_name_patterns	
	pad_power_cut_cell_name_patterns	
IO CORNER CELL	endcap_corner_cell_name_patterns	
IO COVER CELL	cover_cell_name_patterns	
	cover_bump_cell_name_patterns	
BLOCK CELL	block_cell_name_patterns	
	esd_cell_name_patterns	
	ring_cell_name_patterns	
POWER/GROUND	power_pin_name_patterns	*VDD*
	ground_pin_name_patterns	*VSS* *GND*

For GDS and LEF library checking, there are also some necessary configurations for layers should be set before checking.

Library View	Name	Example Value
GDS	gds_boundary_layer	prBoundary
	gds_label_layer_map	
	gds_lef_layer_map	(gdsLayer1 lefLayer1) (gdsLayer2 lefLayer2)
	gds_lib_abutment_layers	
	gds_multi_ground_substrate_layer	PSUB2
	gds_pg_pin_label_layers	
	gds_pg_pin_layers	
	gds_signal_pin_label_layers	
	gds_signal_pin_layers	
	gds_tag_layers	
LEF	lef_lib_abutment_layers	
	lef_pg_pin_layers	
	lef_signal_pin_layers	

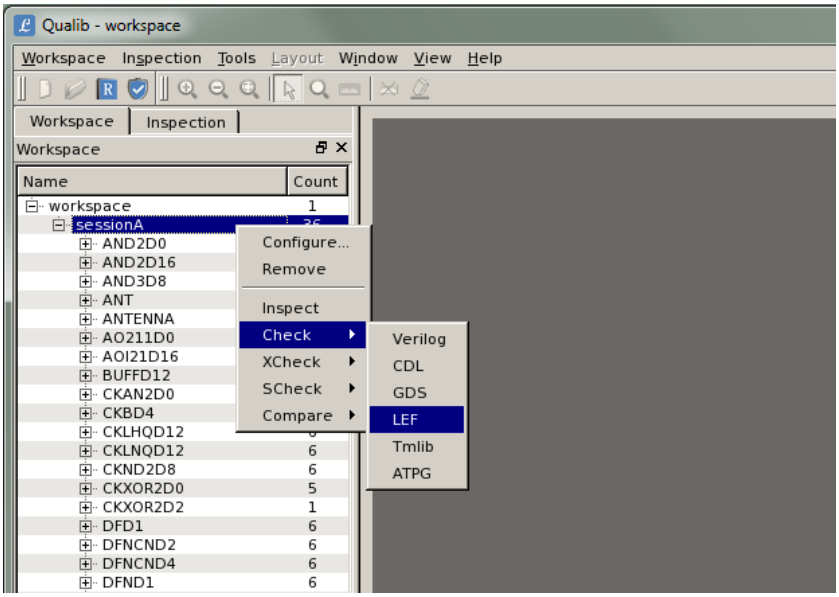
You can also set or display the parameter value in the shell mode.

For example:

```
qualib > set_parameter power_pin_name_patterns "*VDD*"
qualib > set_parameter power_pin_name_patterns
*VDD*
```

Check Library Files

Then you can check, cross-check, self-check, or compare the library files.
Right click the session, and choose the function menu item, and perform the checking.

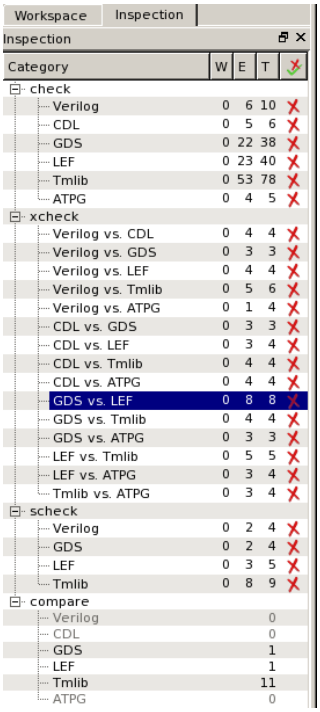



```
qualib > check    -view lef                  -session   sName   [-corners   {…}]
qualib > xcheck   -view_pair {gds lef} -session   sName   [-corner_pair   {…}]
qualib > scheck   -view lef                  -session   sName
qualib > compare -view tmlib   -source_session … -source_corner …   cellA \
                                  -target_session …  -target_corner …   [cellB]
```

Select "***Inspect***" or click  button, it will perform all the check/xcheck/scheck jobs.

```
qualib > inspect   [-session   sessionName]   [-force]
```

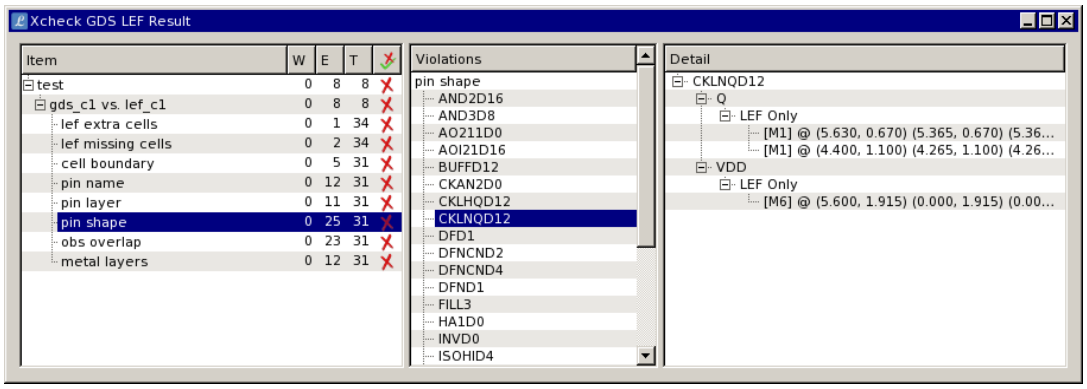
If some related parameters have not been set correctly for "Inspect", it will present warning to users before continue. Or use "inspect -force" command to skip those library checking that are not ready.



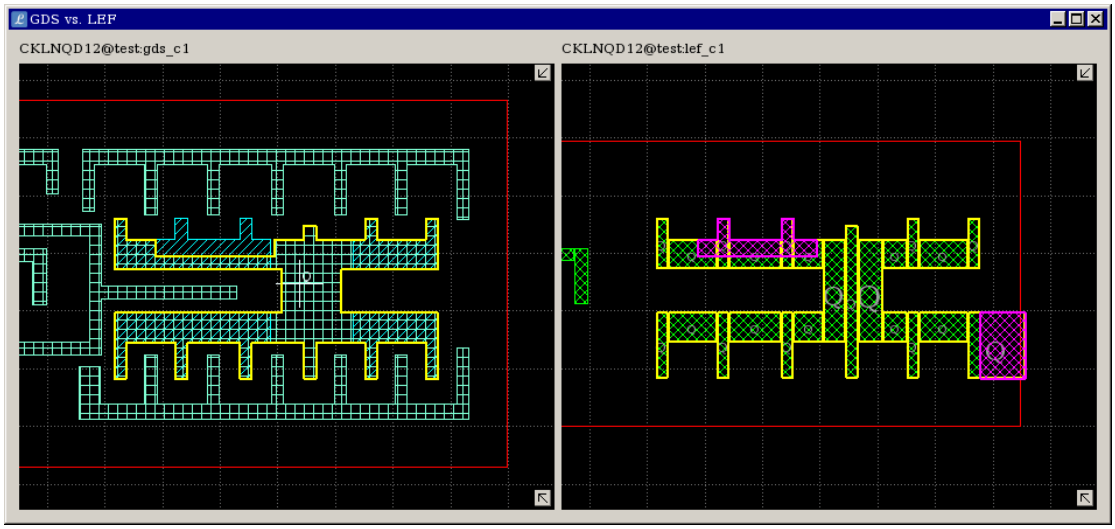
After checking, the inspection summary table will be updated:
"W" indicates the waived check items number;
"E" indicates the error check items number:
"T" indicates the total check items number.

Click the  button to show the failed items only.

Double click each category checking item, the detail result window will pop up.

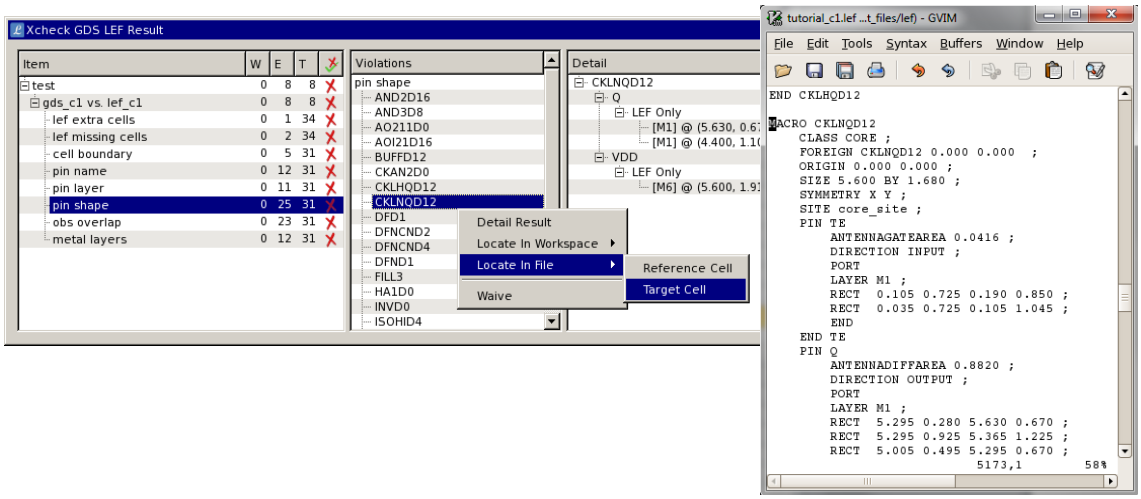


Double click the detail result items, or right click and select "***Detail Info***", the cross probing GUI window will pop up to show the detail layout locations.



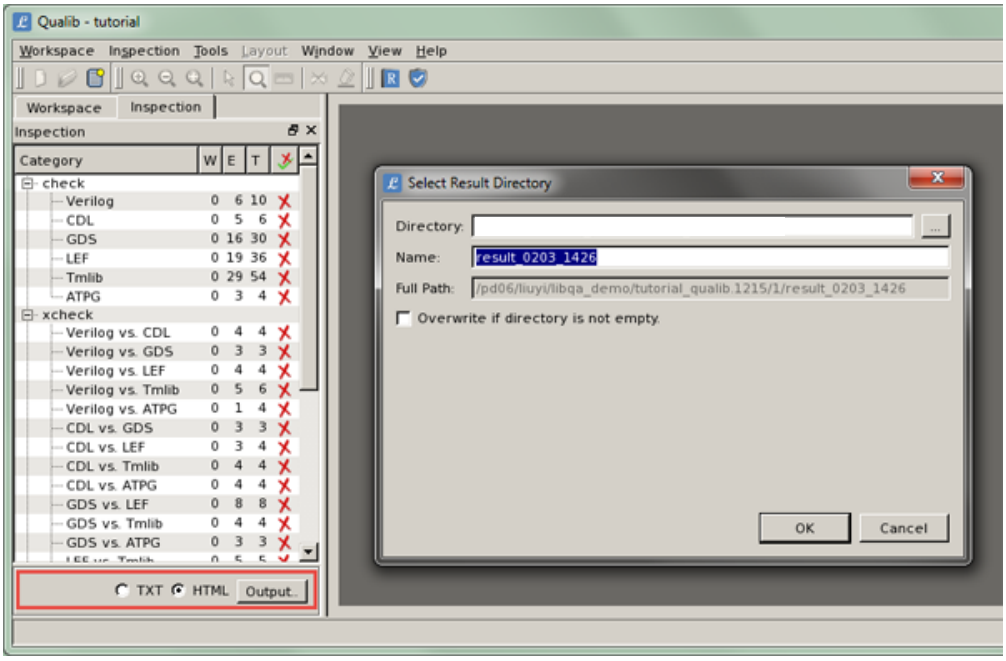
The shapes of selected pin will be drawn in yellow; the difference will be drawn in purple.

You can also right click the result item, locate it in original library file directly.



Output Reports

After checking the libraries, the check result reports can be generated for review.



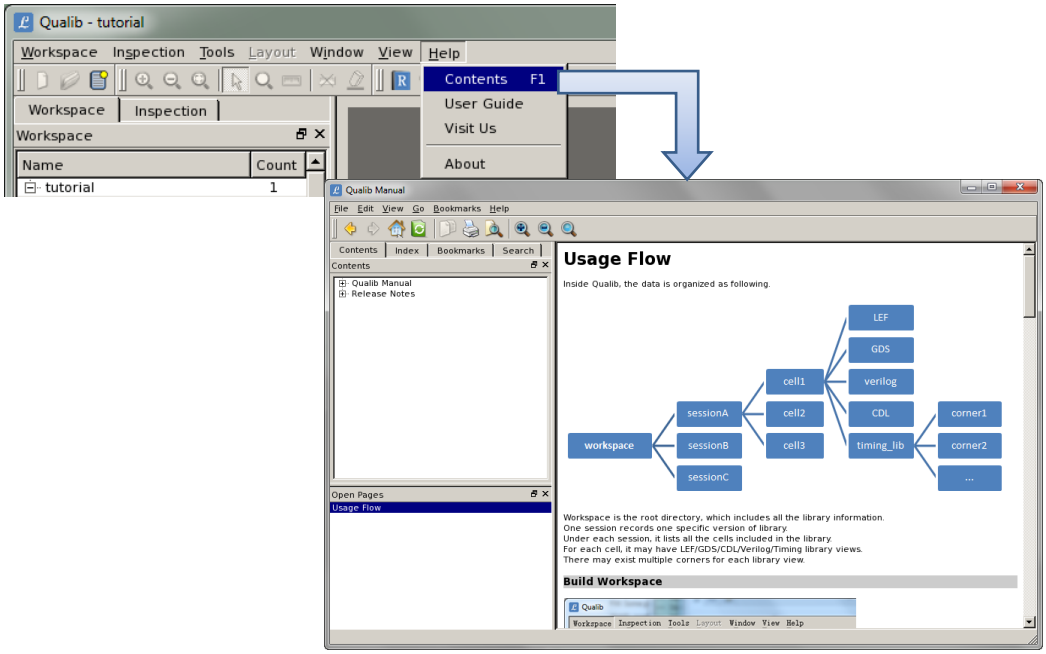
Click "***Output***" button at the bottom of inspection panel, text and HTML format reports can be generated under the user specified directory.

```
qualib > report_inspection_result -dir resultDirectory -format html
qualib > report_inspection_result -dir resultDirectory -format text
```

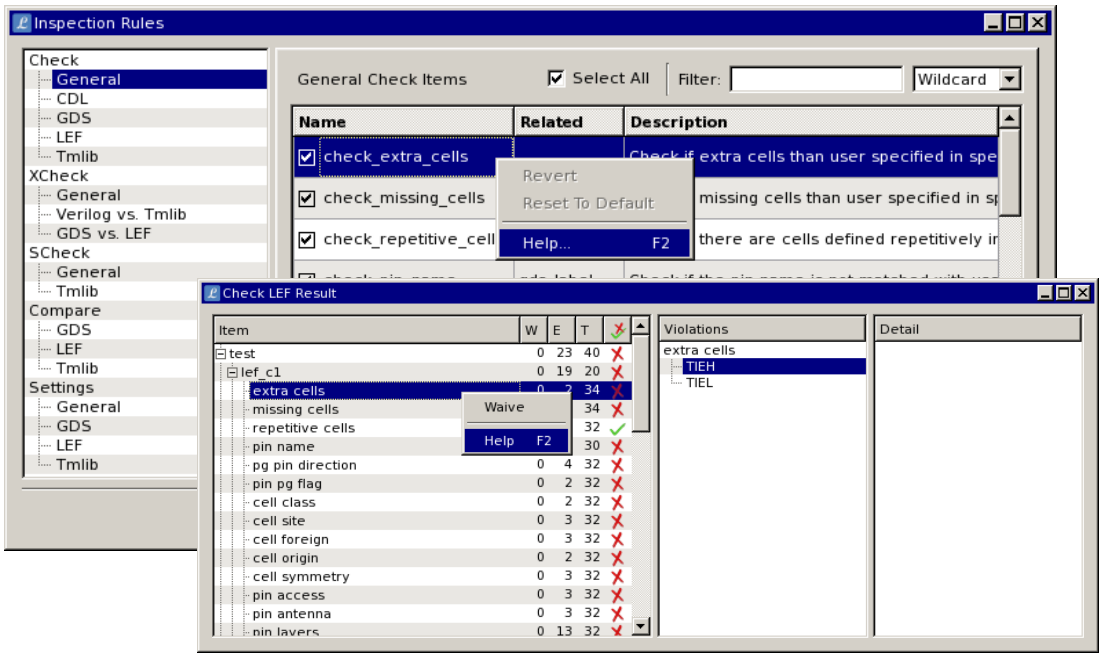
Get Help from Qualib

On-line Help System

The on-line Qualib manual can be invoked from the "***Help*** → **Contents**" menu.
Or directly press the ***F1*** key.



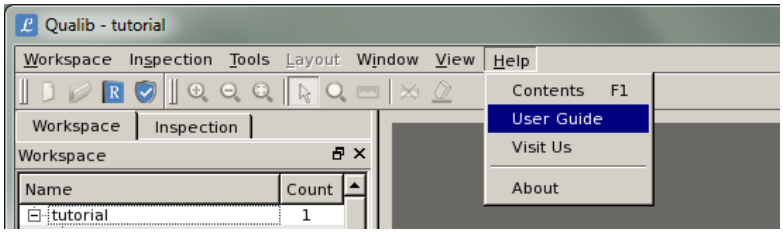
The help can be also triggered from of the inspect rule or check result item.
Or directly press the ***F2*** key on the selected item.



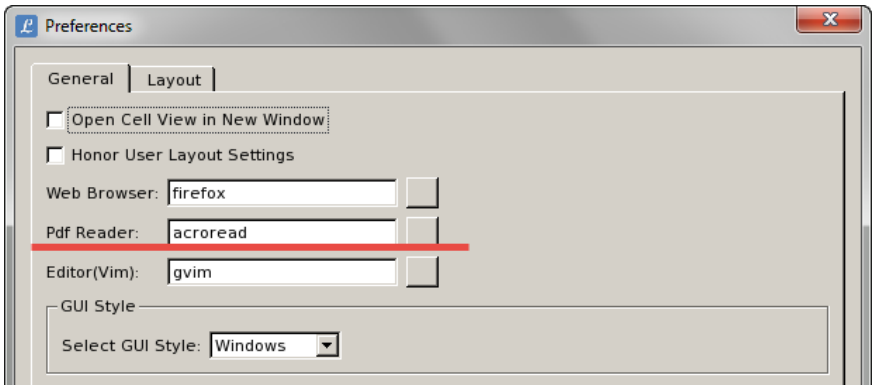
It will pop up Qualib manual to explain the detail meaning of the check item and related commands and parameters.

User Guide

This user guide can be invoked from the "***Help*** → **User Guide**" menu.



To view this PDF document properly, please set the correct PDF reader in preferences. From the "***Tools*** → **Preference…**" menu:



Help Command

User can also type ***help*** command in shell to get the detail explanation of commands.

Qualib > **help** command_name [-verbose]

```
qualib > help create_workspace -verbose
NAME
  create_workspace -
                   Create a workspace for library operations.


SYNOPSIS
  create_workspace
    [-overwrite]
    workspace_path

OPTIONS
  -overwrite       Force to overwrite if the specified directory is not empty.

  workspace_path   The path to create the workspace.


DESCRIPTION
  This command tries to create a workspace in the specified path. If the path
  does not exist, it will create it automatically. If the path already exists
  and is not empty, it will report error, and ask user to specify -overwrite to
  continue.
  It will report error if failed to create the directory or the directory is not
  writable. On success, it will return a string of full path to the workspace.


EXAMPLE

  % create_workspace /home/user/2014.6.1 -overwrite



SEE ALSO
  open_workspace, close_workspace
```

Type ***help_parameter*** command in shell to get the detail explanation of parameters.

Qualib > **help_parameter** parameter_name [-verbose]

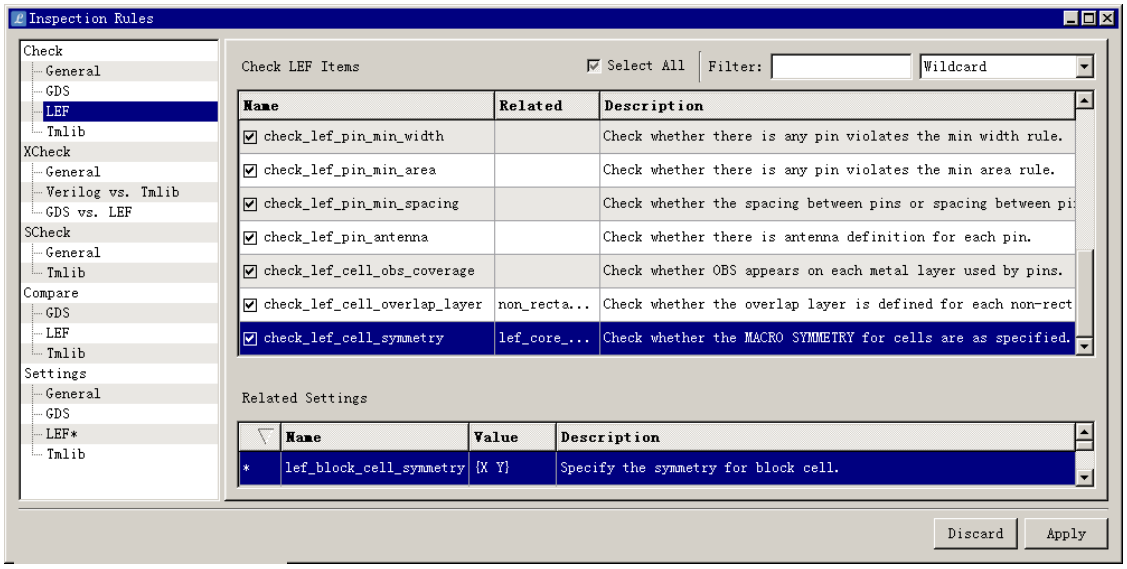
```
qualib > help_parameter check_extra_cells -verbose
NAME
  check_extra_cells -
                   Check if extra cells than user specified in spec.


TYPE
  type = <bool>

DEFAULT VALUE
  default = 'true'

DESCRIPTION
  If this parameter is set to true, the  check  command will check if there are
  extra cells than user specified in spec.
  The spec is specified by the  link_spec  command.


EXAMPLE

    % link_spec -session A ./cell_spec.txt
    % set_parameter check_extra_cells true
    % check -view viewName -session A
    extra cells
      cellX
      cellY



SEE ALSO
  link_spec
```

3
Inspection Rules

This section describes how to set the inspection rules of Qualib.

From the GUI menu ***Inspection → Configure Rules…***, the configuration window will show up. Those selected inspection rules will be performed to check.

Double click the "***Value***" field of each setting, the value can be changed directly. There will be a "*" mark in the beginning, indicating current value has been changed.



Click ***Apply*** button to commit the change, or ***Discard*** button to withdraw the change.

From the GUI menu ***Inspection → Load Rules…***, the parameter settings can be set to pre-defined parameter settings (for *standard cell*, *IP*, and *IO*) or loaded from file.

From the GUI menu ***Inspection → Export Rules…***, current parameter settings can saved into a .tcl script file.

Settings

General Check

In this section, it lists some common settings which will affect the check operations.

Name	Default	Description
abbreviated_xcheck_mode	false	abbreviated xcheck mode in inspection
accumulate_result	false	accumulate result for every inspection result or not
area_error_threshold	0.0001	the error threshold for xchecking cell area
auto_save_load_result	true	save and load the inspection result automatically
block_cell_name_patterns		wildcard name patterns for cells with class BLOCK
core_antenna_cell_name_patterns		wildcard name patterns for CORE antenna cells
core_cell_name_patterns	*	wildcard name patterns for cells with class CORE
core_decap_cell_name_patterns	*DECAP* *DCAP*	wildcard name patterns for decap cells
core_spacer_cell_name_patterns	*FILL*	wildcard name patterns for CORE spacer cells
core_tie_high_low_cell_name_patterns		wildcard name patterns for CORE tie-high/low cells
core_welltap_cell_name_patterns	*TAP*	wildcard name patterns for CORE welltap cells
cover_bump_cell_name_patterns		wildcard name patterns for COVER bump cells
cover_cell_name_patterns		wildcard name patterns for cells with class COVER
decimal_point_tokens	.	tokens for decimal point
disregard_wrong_settings	false	disregard wrong settings or technology in inspection
endcap_corner_cell_name_patterns		wildcard name patterns for cells with class endcap corner
endcap_pre_post_cell_name_patterns		wildcard name patterns for ENDCAP PRE or POST cells
esd_cell_name_patterns		wildcard name patterns for ESD cells
geometry_off_side_threshold	0	geometry off side threshold when xcheck gds & lef
ground_pin_name_patterns	*VSS* *GND*	the GROUND pin name patterns
internal_pin_name_patterns	*:	wildcard name patterns of the GDS/CDL internal pin
negative_sign_token	-	the token for negative sign
non_rectangular_cells		the non-rectangular shaped cells
pad_bonding_cell_name_patterns		wildcard name patterns for PAD cells for bonding
pad_cell_name_patterns		wildcard name patterns for cells with class PAD
pad_power_cut_cell_name_patterns		wildcard name patterns for power cut PAD cells
pad_power_ground_cell_name_patterns		wildcard name patterns for PAD POWER cells
pad_spacer_cell_name_patterns		wildcard name patterns for PAD SPACER cells
pg_pin_direction	inout	the power/ground pin direction
positive_sign_token	+	the token for positive sign
power_pin_name_patterns	*VDD*	the POWER pin name patterns
preferred_pin_map		pin map for comparing cells with different name styles
result_tree_branch_upper_bound	3000	the upper bound of tree branches in detail result
ring_cell_name_patterns		wildcard name patterns for cells with class RING
scan_cell_name_patterns		wildcard name patterns for scan cells

GDS Check

In this section, it lists some common settings which will affect GDS view checks.

Name	Default	Description
gds_boundary_layer	prBoundary	the PR boundary layer in GDS
gds_bus_bit_chars	[]	the bus bit chars for GDS
gds_close_to_boundary_dont_check_layers		the don't check layers for GDS close to boundary check
gds_dummy_layers		the dummy layers in GDS
gds_geometry_search_depth	20	the geometry search depth in GDS design
gds_label_layer_map		the layer map between labels to pin for GDS design
gds_layer_hierarchy		the layers on specified hierarchical level
gds_lef_layer_map		the layer map between GDS design and LEF design
gds_lef_strict_pin_layer_check	false	if cross check GDS and LEF pin layers strictly
gds_lef_strict_pin_shape_check	false	if cross check GDS and LEF pin shapes strictly
gds_lib_abutment_layers		the GDS abutment layers
gds_lib_abutment_reference_cell		the reference cell for GDS abutment rule check
gds_lib_leaf_cell_file		the file name containing the GDS leaf cells
gds_multi_ground_substrate_layer	PSUB2	the substrate layer name if multi GROUND in GDS
gds_off_boundary_dont_check_layers		the don't check layers for GDS off boundary check
gds_pg_pin_label_layers		the layers should be used by PG pin label in GDS
gds_pg_pin_layers		the layers should be used by PG pin in GDS
gds_pin_obs_spacing	0	the spacing between pin and OBS in GDS
gds_signal_pin_label_layers		the layers should be used by signal pin label in GDS
gds_signal_pin_layers		the layers should be used by signal pin in GDS
gds_tag_content		the tag content in GDS
gds_tag_layers		the tag layers in GDS
gds_tag_origin	(0, 0)	the tag origin in GDS design
gds_trace_for_pg_pin	false	if trace geometries for power/ground pin in GDS
gds_trace_for_signal_pin	false	if trace geometries for signal pin in GDS

LEF Check

In this section, it lists some common settings which will affect LEF view checks.

Name	Default	Description
lef_antenna_dont_check_pins		the don't check pins for LEF pin antenna check
lef_block_cell_symmetry	{X Y R90}	the symmetry for BLOCK cell
lef_cell_foreign	(0, 0)	the default cell foreign
lef_cell_origin	(0, 0)	the default cell origin
lef_core_cell_symmetry	{X Y}	the symmetry for CORE cell
lef_enable_pin_width_check	true	check LEF pin width during touching boundary check
lef_lib_abutment_layers		the LEF abutment layers

Inspection Rules

Name	Default	Description
lef_lib_abutment_reference_cell		the reference cell for LEF lib cell abutment rule check
lef_pad_cell_symmetry	{X_Y_R90}	the symmetry for PAD cell
lef_pg_pin_layers		the layers should be used by PG pin in LEF
lef_signal_pin_layers		the layers should be used by signal pin in LEF
lef_unlimited_routing_layers		the LEF routing layers not need to be covered by OBS

Timing Lib Check

In this section, it lists some common settings which will affect Timing library view checks.

Name	Default	Description
tmlib_bus_downto	true	the preferred bus order for bus check
tmlib_cap_constraint_dont_check_cell_name_patterns		cells not be checked the cap constraint
tmlib_constraint_value_range	[-inf, inf]	the valid constraint value range
tmlib_delay_input_threshold_pct	50	the rise/fall edge input threshold for delay calculation in percentage
tmlib_delay_output_threshold_pct	50	the rise/fall edge output threshold for delay calculation in percentage
tmlib_delay_relative_tolerance	0.05	the relative tolerance for ccs and nldm delay
tmlib_delay_value_range	[-inf, inf]	the valid delay value range
tmlib_file_name_format		the timing library file name format on PVT
tmlib_internal_power_conflict_threshold	1e-05	the cell internal power conflict threshold
tmlib_internal_power_value_range	[-inf, inf]	the valid power value range
tmlib_leakage_power_dont_check_cell_name_patterns		cells not be checked the leakage power
tmlib_leakage_power_range	(0, 1e-05)	the valid cell leakage power range
tmlib_library_name_format		the timing library lib name format on PVT
tmlib_lowest_but_bit	0	the lowest bus bit for bus checking
tmlib_max_capacitance_range	(0, inf)	the max capacitance legal range
tmlib_max_transition_range	(0, 5e-09)	the max transition legal range
tmlib_monotonicity_threshold	1e-05	the threshold for the timing monotonicity check
tmlib_operating_conditions_name_format		the timing library operating conditions name format on PVT
tmlib_pin_capacitance_range	(0, 5e-12)	the valid pin capacitance range
tmlib_power_down_function_verification	false	verify output pin attr ‘power_down_function’
tmlib_power_table_index_step_growth_rate	2	the max step size for the power table index
tmlib_slew_derate_threshold	0.5	the desired slew derate value of timing library
tmlib_slew_lower_threshold_pct	30	the desired slew lower threshold in percentage
tmlib_slew_upper_threshold_pct	70	the desired slew upper threshold in percentage
tmlib_table_index_conflict_skip_different_dimension_tables	false	if skip the different dimension model in table index conflict checking
tmlib_table_index_conflict_threshold	1e-05	the table index conflict threshold
tmlib_timing_table_index_step_growth_rate	5	the max step size for the timing table index
tmlib_transition_relative_tolerance	0.1	the relative tolerance for ccs and nldm transition
tmlib_transition_value_range	[-inf, inf]	the valid transition value range

Check Rules

General Check

In this section, it lists the general checking rule items for each library view.

Checking operations which have been selected will be performed.

Name	Description
check_cell_class	cell class of the specified cells, only effective for LEF
check_extra_cells	if there are extra cells compared with the specified spec
check_missing_cells	if there are missing cells compared with the specified spec
check_pg_pin_direction	if the power/ground pin direction is not the specified type
check_pin_bus_name	if the pin named like a bus bit is not really a bus bit
check_pin_name	if the pin name is not matched with those specified in spec
check_pin_pg_flag	check power/ground flag of all pins
check_repetitive_cells	if there are multiple cell definitions in different files of same corner

LEF Check

In this section, it lists the every checking rule item for LEF library files.

Checking operations which have been selected will be performed.

Name	Description
check_lef_cell_foreign	if the cell foreign is not at the specified coordinates
check_lef_cell_obs_coverage	if OBS appears on each pin metal layer
check_lef_cell_origin	if the cell origin is not at the specified coordinates
check_lef_cell_overlap_layer	if the overlap layer is not defined for non-rectangular cells
check_lef_cell_site	if the site of cell is not defined in tech
check_lef_cell_symmetry	if the cell symmetry is not defined as specified
check_lef_close_to_boundary	if any pin shape/OBS is too close to the PR boundary
check_lef_layer_pin_existence	if pin doesn’t exist on the user specified layers
check_lef_lib_cell_abutment	if any core cell violates the abutment rule
check_lef_off_boundary	if all the pin shapes/OBS are out of the PR boundary
check_lef_pin_access	if all the pins are not accessible for routing
check_lef_pin_antenna	if there is no antenna definition for each pin
check_lef_pin_grid_alignment	if all the pins are not on the manufacturing grid
check_lef_pin_layers	if signal pins and P/G pins are on the specified layers
check_lef_pin_min_area	if there is any pin violating the min area rule
check_lef_pin_min_spacing	if there is any pin violating the min spacing rule
check_lef_pin_touch_boundary	if signal pins in block cell don't touch PR boundary

GDS Check

In this section, it lists the every checking rule item for GDS library files.

Checking operations which have been selected will be performed.

Name	Description
check_gds_boundary_origin	if the boundary origin is not at (0, 0)
check_gds_close_to_boundary	if metal layer shapes are too close to the PR boundary
check_gds_grid_alignment	if any shape on metal layers is not on the manufacturing grid
check_gds_label_conflict	if there are different labels on the same shape
check_gds_label_geometry	if the label height is 0.0
check_gds_label_layers	if the pin labels are not on the specified layers
check_gds_label_shape_overlap	if any label has no corresponding shape
check_gds_layer_hierarchy	if the specified layers don't exist on specified hierarchy level
check_gds_layer_pin_existence	if pin doesn’t exist on the user specified layers
check_gds_lib_cell_abutment	if any core cell violates the abutment rule
check_gds_lib_conflicting_cells	if any cell used in a corner is not unique
check_gds_lib_database_unit	if the database unit conflict in gds files in a corner
check_gds_lib_layer_map_integrity	if the gds file related layers have been defined completely
check_gds_lib_missing_leaf_cells	if any leaf cell in cell list doesn't appear in gds file in a corner
check_gds_multi_boundary	if there are more than one boundaries on the top level
check_gds_multi_ground_substrate_layer	if the specified substrate layer doesn’t exist
check_gds_off_boundary	if all the shapes/texts are out of the cell boundary
check_gds_pin_layers	if the pins are not on the specified layers
check_gds_tag_content	if the content of a tag is not in the user specified format
check_gds_tag_origin	if the tag origin is not at specified location

Timing Lib Check

In this section, it lists the every checking rule item for Timing library files.

Checking operations which have been selected will be performed.

Name	Description
check_tmlib_area	if cell area is redefined or undefined
check_tmlib_bus	if bus definition is not correct
check_tmlib_ccs_index	if ccs timing indexes are illegal
check_tmlib_ccs_nldm_timing_correlation	if timing value relative differences in ccs & nldm are larger than the tolerance
check_tmlib_condition	if “when” and “sdf_cond” of group are not defined equally
check_tmlib_constraint_conflict	if setup/hold constraint groups are not defined correctly
check_tmlib_default_operating_condition	if default operating condition is not as user specified
check_tmlib_default_same_as_nominal	if default operating condition is not same as nominal
check_tmlib_file_name	if file name is not matched with user specified format

Inspection Rules

Name	Description
check_tmlib_footprint	if cell_footprint is redefined or undefined
check_tmlib_internal_power_arc_presence	if internal power arcs are not completely defined
check_tmlib_internal_power_conflict	if the sum of rise and fall power value is less than user specified threshold
check_tmlib_internal_power_table_presence	if tables are not completely defined for a power group
check_tmlib_leakage_power	if defined cell leakage power value is not within user specified range
check_tmlib_lib_slew_delay_definition	if the slew definition of timing library is not desired value
check_tmlib_lib_unit_definition	if unit is not defined correctly in libraries in corner
check_tmlib_library_name	if the lib name is not matched with user specified format
check_tmlib_library_name_in_corner	if the lib name is not same as other lib names in corner
check_tmlib_noise_table_presence	if noise table is not completely defined in ccsn group
check_tmlib_operating_conditions_name	if the operating condition name is not matched with user specified format
check_tmlib_pg_pin_with_voltage_name	if voltage name is not defined on voltage maps
check_tmlib_pin_with_capacitance	if capacitance is defined for each input pin or is not within user specified range
check_tmlib_pin_with_max_capacitance	if max_capacitance is not defined for an output pin or is not larger than 0
check_tmlib_pin_with_max_transition	if max_transition is not defined for an input pin
check_tmlib_pin_with_min_capacitance	if min_capacitance is not defined for an output pin
check_tmlib_pin_with_min_pulse_width	if min_pulse_width group is not defined for a clock pin
check_tmlib_pin_with_power_down_function	if ‘power_down_function’ attribute is not defined on output/inout pin
check_tmlib_pin_with_related_ground_pin	if related ground pin is not defined on pin
check_tmlib_pin_with_related_power_pin	if related power pin is not defined on pin
check_tmlib_table_index_conflict	if the indexes of timing() and internal_power() group tables are not identical
check_tmlib_table_index_step_growth_rate	if the step growth rate of the table index is too big
check_tmlib_table_value	if any value of timing table is not in the specified range
check_tmlib_timing_arc_presence	if the timing arcs are not completely defined
check_tmlib_timing_monotonicity_with_load	if the output delay/trans is not monotone with the output load
check_tmlib_timing_monotonicity_with_slew	if the output delay/trans is not monotone with the input slew
check_tmlib_timing_table_presence	if tables are not completely defined for a timing group
check_tmlib_voltage_map_definition	if voltage map is not defined or voltage value 0 not appear

CDL Check

In this section, it lists the every checking rule item for CDL library files.

Checking operations which have been selected will be performed.

Name	Description
check_cdl_global_nodes	if global nodes have been defined

ATPG Check

In this section, it lists the every checking rule item for ATPG library files.

Checking operations which have been selected will be performed.

Name	Description
check_atpg_non_scan_model	if cell non_scan_model is not correct
check_atpg_scan_definition_presence	if scan definition is missing or redundant

X-Check Rules

General Cross-Check

In this section, it lists the general cross-checking rule item between library view files.

Checking operations which have been selected will be performed.

Name	Description
xcheck_cell_area	cross check cell area of the specified library views
xcheck_cell_boundary	cross check cell boundary of the specified library views
xcheck_extra_cells	cross check extra cells in the right hand side view
xcheck_missing_cells	cross check missing cells in the right hand side view
xcheck_pin_direction	cross check pin direction of the specified library views
xcheck_pin_name	cross check pin name of the specified library views

Verilog vs. Timing Lib Cross-Check

In this section, it lists the every cross-checking rule item between verilog netlist files and timing liberty files.

Checking operations which have been selected will be performed.

Name	Description
xcheck_verilog_tmlib_bus	if any bus pin in timing lib is not matched with Verilog
xcheck_verilog_tmlib_pin_function	if any pin function in timing lib is not matched with Verilog
xcheck_verilog_tmlib_timing_arc	if any timing arc in timing lib is not matched with Verilog
xcheck_verilog_tmlib_timing_constraint	if any timing check in timing lib is not matched with Verilog

GDS vs. LEF Cross-Check

In this section, it lists the every cross-checking rule item between GDS and LEF files.

Checking operations which have been selected will be performed.

Name	Description
xcheck_gds_lef_metal_layers	if metal layers in LEF design and GDS design are not matched
xcheck_gds_lef_obs_overlap	if the GDS metal is not totally covered by LEF OBS plus LEF pin shapes
xcheck_gds_lef_pin_layer	if the LEF pin layers are not covered by GDS pin layers
xcheck_gds_lef_pin_shape	if the LEF pin shapes are not covered by corresponding GDS pin shapes

S-Check Rules

General Self-Check

In this section, it lists the general self-checking rule item between library view files in different corners of the same session.

Checking operations which have been selected will be performed.

Name	Description
scheck_cell_area	check if cell area is not matched with the reference corner for a specified view
scheck_cell_boundary	check if cell boundary is not matched with the reference corner for a specified view
scheck_extra_cells	check if extra cells than the reference corner for a specified view
scheck_missing_cells	check if missing cells than the reference corner for a specified view
scheck_pin_direction	check if pin direction is not matched with the reference corner for a specified view
scheck_pin_name	check if pin name is not matched with the reference corner for a specified view

Timing Lib Self-Check

In this section, it lists the every self-checking rule item between timing liberty files under different corners of the same session.

Checking operations which have been selected will be performed.

Name	Description
scheck_tmlib_leakage_power_group	check if any leakage power group is not matched with the reference corner
scheck_tmlib_pin_function	check if any pin function is not matched with the reference corner
scheck_tmlib_power_arc	check if any power arc is not matched with the reference corner
scheck_tmlib_timing_arc	check if any timing arc is not matched with the reference corner

Compare Rules

GDS Comparing

In this section, it lists all the comparison rules can be performed between two GDS library cells. Checking operations which have been selected will be performed.

Parameter Name	Description
compare_gds_cell_layout	compare the layout of two GDS library cells

LEF Comparing

In this section, it lists all the comparison rules can be performed between two LEF library cells. Checking operations which have been selected will be performed.

Parameter Name	Description
compare_lef_cell_layout	compare the layout of two LEF library cells

Timing Lib Comparing

In this section, it lists all the comparison rules can be performed between two timing liberty files. Checking operations which have been selected will be performed.

Parameter Name	Description
compare_tmlib_area	compare the area of two liberty cells
compare_tmlib_hold_constraint	compare hold constraint values of two liberty cells
compare_tmlib_internal_power	compare internal power values of two liberty cells
compare_tmlib_leakage_power	compare the leakage power of two liberty cells
compare_tmlib_pin_capacitance	compare the input pin capacitance of two liberty cells
compare_tmlib_pin_max_capacitance	compare output pin max capacitance of two liberty cells
compare_tmlib_pin_max_fanout	compare output pin max fanout of two liberty cells
compare_tmlib_pin_max_transition	compare input pin max transition of two liberty cells
compare_tmlib_pin_min_capacitance	compare output pin min capacitance of two liberty cells
compare_tmlib_recovery_constraint	compare recovery constraint values of two liberty cells
compare_tmlib_removal_constraint	compare removal constraint values of two liberty cells
compare_tmlib_setup_constraint	compare setup constraint values of two liberty cells
compare_tmlib_timing_delay	compare timing delay values of two liberty cells
compare_tmlib_timing_transition	compare timing transition values of two liberty cells

4 Checking

This section describes the library files checking functionalities.

General

There are some general checking items for each type of library view.

cell_class

Check Item: check if cell class is specified correctly.

Related Parameter: **check_cell_class**
core_cell_name_patterns
core_antenna_cell_name_patterns
core_decap_cell_name_patterns
core_spacer_cell_name_patterns
core_tie_high_low_cell_name_patterns
core_welltap_cell_name_patterns
block_cell_name_patterns
ring_cell_name_patterns
esd_cell_name_patterns
pad_cell_name_patterns
pad_bonding_cell_name_patterns
pad_power_cut_cell_name_patterns
pad_power_ground_cell_name_patterns
pad_spacer_cell_name_patterns
cover_cell_name_patterns
cover_bump_cell_name_patterns
endcap_pre_post_cell_name_patterns
endcap_corner_cell_name_patterns

extra_cell

Check Item: check if there are extra cells compared with user specified in spec.

Related Parameter: **check_extra_cells**

missing_cell

Check Item: check if there are missing cells compared with user specified in spec.

Related Parameter: **check_missing_cells**

pg_pin_direction

Check Item: check if the power/ground pin direction is not specified correctly.

The specified P/G pin direction should be defined by parameter *pg_pin_direction*.

Related Parameter: **check_pg_pin_direction**
pg_pin_direction
power_pin_name_patterns
ground_pin_name_patterns

pin_bus_name

Check Item: check if the pin named like a bus bit is not really a bus bit.

Related Parameter: **check_pin_bus_name**
gds_bus_bit_chars

pin_name

Check Item: check if the pin name is not matched with user specified in spec.

Related Parameter: **check_pin_name**
internal_pin_name_patterns
gds_label_layer_map

pin_pg_flag

Check Item: check if the power/ground flag is specified correctly for P/G pins.

Related Parameter: **check_pin_pg_flag**
power_pin_name_patterns
ground_pin_name_patterns

repetitive_cells

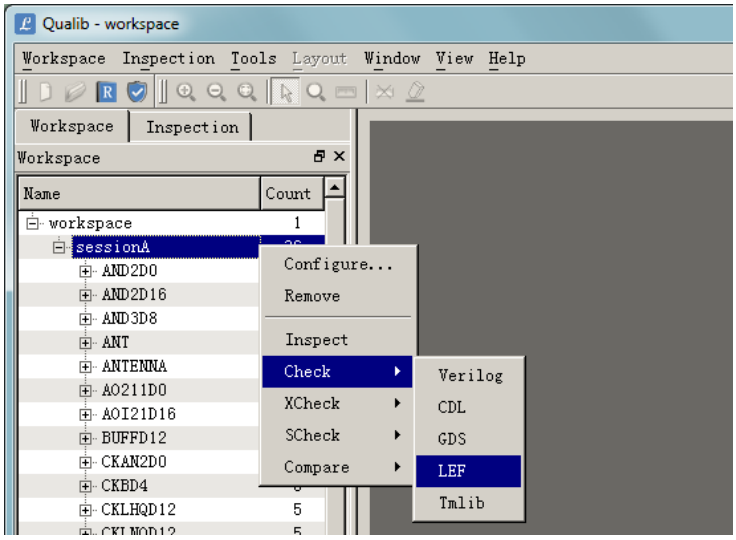
Check Item: check if there are cells defined repetitively in different files under the same corner.

Related Parameter: **check_repetitive_cells**

LEF

To check the LEF library files, click on the GUI menu ***Inspection → Check → LEF***.

Or just right click the session name, and choose ***Check → LEF***.




```
qualib > check -view lef  -session  sessionName
```

According to the control parameter setting of "LEF Check", enabled checking items will be performed.

cell_foreign

Check Item: check if the cell foreign is not at the specified coordinates.

```
MACRO AD1HSV1
  CLASS CORE ;
  ORIGIN 0 0 ;
  FOREIGN AD1HSV1 0 0 ;
  SIZE 5.04 BY 1.26 ;
  SYMMETRY X Y ;
  SITE CoreSite ;
  PIN A
```

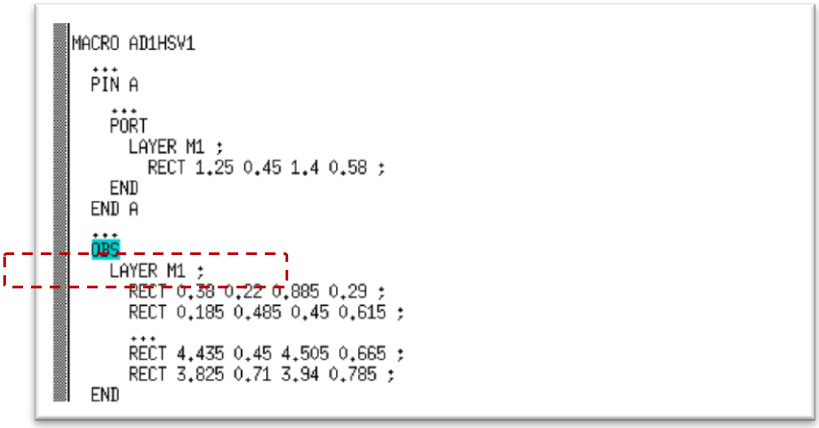
The specified coordinates should be defined by parameter *lef_cell_foreign*.

Related Parameter: **check_lef_cell_foreign**
lef_cell_foreign

cell_obs_coverage

Check Item: check if any metal layer is not covered by OBS.

The LEF layers specified by the parameter *lef_unlimited_routing_layers* will not be checked.



Related Parameter: **check_lef_cell_obs_coverage**
lef_unlimited_routing_layers

cell_origin

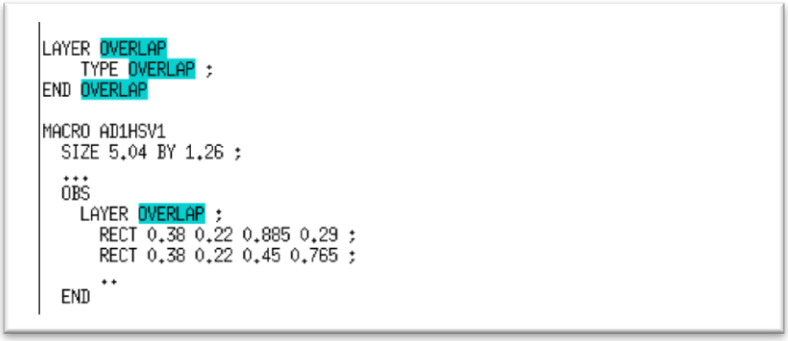
Check Item: check if the cell origin is not at the specified coordinates.

The specified coordinates should be defined by parameter *lef_cell_orign*.

Related Parameter: **check_lef_cell_origin**
lef_cell_origin

cell_overlap_layer

Check Item: check if no overlap layer is not defined for a non-rectangular cell.



The non-rectangular cells are defined by the parameter *non_rectangular_cells*.

Related Parameter: **check_lef_overlap_layer**
non_rectangular_cells

cell_site

Check Item: check if the cell site is not defined in the tech LEF.

```
MACRO AD1HSV1
  CLASS CORE ;
  ORIGIN 0 0 ;
  FOREIGN AD1HSV1 0 0 ;
  SIZE 5.04 BY 1.26 ;
  SYMMETRY X Y ;
  SITE CoreSite ;
  PIN A
```

Related Parameter: **check_lef_cell_site**

cell_symmetry

Check Item: check if the cell symmetry is not defined as specified.

```
MACRO AND2HSV1
  CLASS CORE ;
  ORIGIN 0 0 ;
  FOREIGN AND2HSV1 0 0 ;
  SIZE 0.84 BY 1.26 ;
  SYMMETRY X Y ;
  SITE CoreSite ;
```

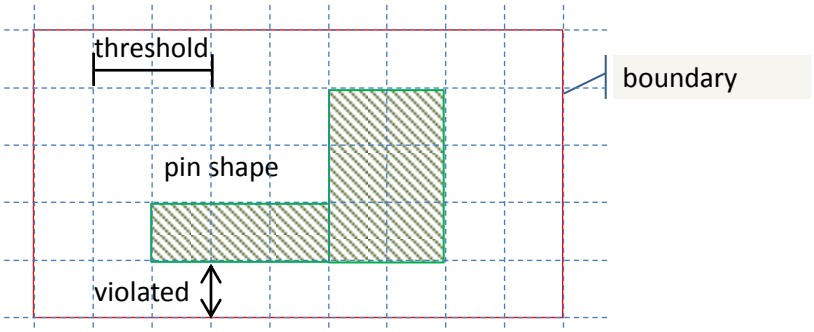
The block cells are specified by the parameter *block_cell_name_patterns*.
The core cells are specified by the parameter *core_cell_name_patterns*.
The pad cells are specified by the parameter *pad_cell_name_patterns*.

Block cell symmetry is specified by the parameter *lef_block_cell_symmetry*.
Standard cell symmetry is specified by the parameter *lef_core_cell_symmetry.*
PAD cell symmetry is specified by the parameter *lef_pad_cell_symmetry*.

Related Parameter: **check_lef_cell_symmetry**
lef_block_cell_symmetry
lef_core_cell_symmetry
lef_pad_cell_symmetry
block_cell_name_patterns
core_cell_name_patterns
pad_cell_name_patterns

close_to_boundary

Check Item: check if any shape/OBS is too close to the PR boundary, except for the power/ground pins.



If the distance between shape/OBS and the PR boundary is less than the specified threshold, defined by ½ layer min spacing, the shape/OBS will be reported as too close to the PR boundary.

The block cells are specified by the parameter *block_cell_name_patterns*.
The pad cells are specified by the parameter *pad_cell_name_patterns*.
For the BLOCK and PAD cells, this check will be skipped.

Related Parameter: **check_lef_close_to_boundary**
power_pin_name_patterns
ground_pin_name_patterns
block_cell_name_patterns
pad_cell_name_patterns

layer_pin_existence

Check Item: check if signal/PG pin shapes are not defined on the correct layers.

This will check if signal and P/G pin shapes exist on the layers specified by the parameters: *lef_signal_pin_layers* and *lef_pg_pin_layers*. For the pins on some layer, if there are some missing pins not defined on some layer, the layer will be reported.

The power pin names will be defined by the parameter *power_pin_name_patterns*.
The ground pin names will be defined by the parameter *ground_pin_name_patterns*.

Related Parameter: **check_lef_layer_pin_existence**
power_pin_name_patterns
ground_pin_name_patterns
lef_signal_pin_layers
lef_pg_pin_layers

lib_cell_abutment

Check Item: check if any core cell violates the abutment rule.

The check is intended to ensure that the core cell can be placed adjacent to each other in rows or columns, fitting to their neighbors. The core cells are specified by the parameters *core_cell_name_patterns*, The BLOCK and PAD cells are specified by the parameters *block_cell_name_patterns*, and *pad_cell_name_patterns*.

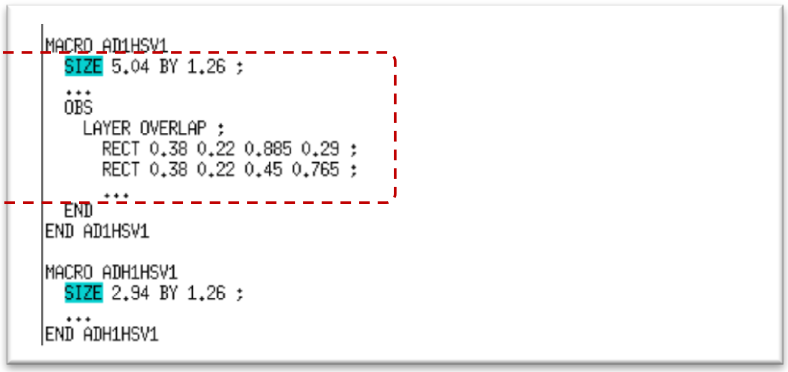
The reference cell is specified by the parameter *lef_lib_abutment_reference_cell*. The layers for LEF lib cell abutment rule check is specified by the parameter *lef_lib_abutment_layers*. Here the reference cell should have the origin/foreign specified by the parameters *lef_cell_origin* and *lef_cell_foreign*. Otherwise, error will be reported for reference cell. If reference cell is not specified, the cell with minimum area will be chosen as reference cell. And this cell MUST has signal pin.

For example, the power and ground lines of a cell should have the same width and occur at the same position to ensure that enough current can be provided. Otherwise, it will cause the LVS/DRC problems.

Related Parameter: **check_lef_lib_cell_abutment**
lef_lib_abutment_reference_cell
lef_lib_abutment_layers
block_cell_name_patterns
core_cell_name_patterns
pad_cell_name_patterns
lef_cell_foreign
lef_cell_origin

off_boundary

Check Item: check if any pin shape or OBS is out of the PR boundary.



Normal cells have rectilinear PR boundary, but for non-rectangular cells, the PR boundary is defined by the OVERLAP layer.

In general, the power/ground pins of a standard cell are out of the boundary. So power/ground pins will only be checked for IP and PAD cells, which are specified by parameters *block_cell_name_patterns* and *pad_cell_name_patterns*.

The power pin names will be defined by the parameter *power_pin_name_patterns*. The ground pin names will be defined by the parameter *ground_pin_name_patterns*.

Related Parameter: **check_lef_off_boundary**
power_pin_name_patterns
ground_pin_name_patterns
block_cell_name_patterns
pad_cell_name_patterns

pin_access

Check Item: check if any pin is not accessible for routing.

If a pin (including P/G pin) cannot be accessed by metal or via, it will be reported.

-- If there is no pin shape in a pin, it will be reported as "No Pin Shape".

-- If there are pin shapes but there is no access point, the pin will be reported as "No Access Point". A pin shape has access point when there are locations fully overlapped by metal with size of a half layer width multiply by layer width. These locations will not cause DRC issues with other physical geometries.

-- If pin shapes are totally blocked by OBS, the pin will be reported as "OBS blocked". According to LEF, routing can connect to a port on the same layer if the routing does not cross any obstruction by more than a distance of the total of minimum width plus minimum spacing before reaching the pin. This is because the port geometry is known to be real and any obstruction less than a distance of minimum width plus minimum spacing away from the port is not a real obstruction. If the pin is more than minimum width plus minimum spacing away from the obstruction edge, the router can only route to the pin from the layer above or below using a via. If there is no valid via or dropping via will create DRC/LVS violation, this pin is totally blocked by OBS and cannot be accessed.

Please refer to “Macro Obstruction Statement” in LEF document for details.

Related Parameter: **check_lef_pin_access**

pin_antenna

Check Item: check if there is no antenna definition for a pin.

```
[PIN pinName
    [ANTENNAPARTIALMETALAREA value [LAYER layerName] ;] ...
    [ANTENNAPARTIALMETALSIDEAREA value [LAYER layerName] ;] ...
    [ANTENNAPARTIALCUTAREA value [LAYER layerName] ;] ...
    [ANTENNADIFFAREA value [LAYER layerName] ;] ...
    [ANTENNAMODEL {OXIDE1 | OXIDE2 | OXIDE3 | OXIDE4} ;] ...
    [ANTENNAGATEAREA value [LAYER layerName] ;] ...
    [ANTENNAMAXAREACAR value LAYER layerName ;] ...
    [ANTENNAMAXSIDEAREACAR value LAYER layerName ;] ...
    [ANTENNAMAXCUTCAR value LAYER layerName ;] ...
END pinName]
```

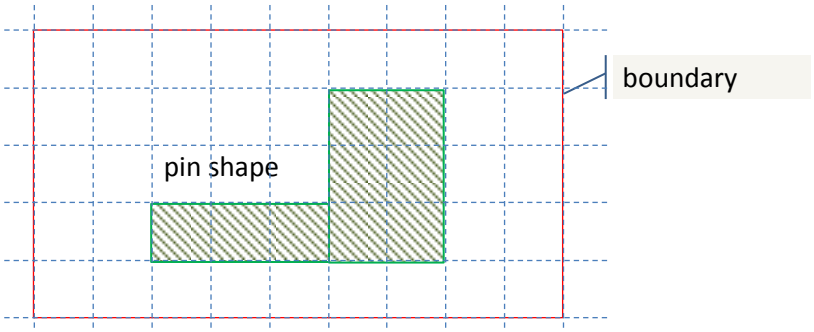
If the pin has none of the antenna definitions above, it will be reported.

The dont-check pins will be skipped to check, which are specified by the parameter *lef_antenna_dont_check_pins*.

Related Parameter: **check_lef_pin_antenna**
lef_antenna_dont_check_pins

pin_grid_alignment

Check Item: check if any pin shape is not on the manufacturing grid.

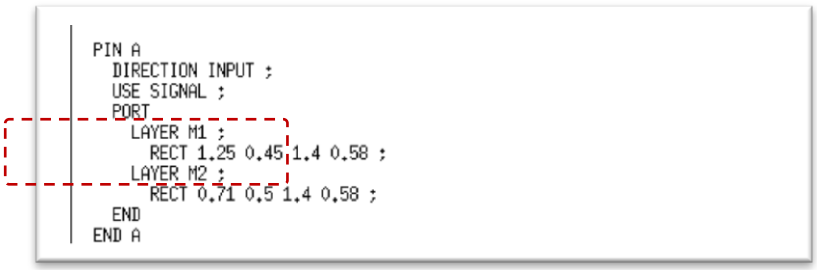


Manufacturing grid is defined in LEF by MANUFACTURINGGRID keyword.

Related Parameter: **check_lef_pin_grid_alignment**

pin_layers

Check Item: check if any pin is not on the user specified layers.



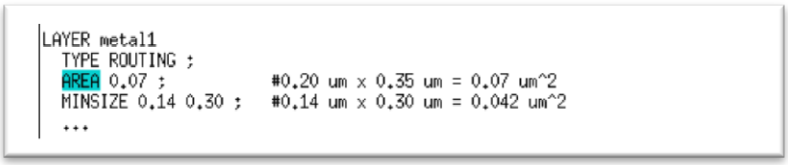
The user specified layers will be defined by the parameter *lef_signal_pin_layers* and *lef_pg_pin_layers* for signal pins and power/ground pins.

The power pin names will be defined by the parameter *power_pin_name_patterns*. The ground pin names will be defined by the parameter *ground_pin_name_patterns*.

Related Parameter: **check_lef_pin_layers**
lef_signal_pin_layers
lef_pg_pin_layers
power_pin_name_patterns
ground_pin_name_patterns

pin_min_area

Check Item: check if any pin violates the min area constraint.



The polygons of pin shapes must have a minimum area (defined by AREA keyword), or a rectangle (defined by MINSIZE keyword) must be able to fit within the polygon.

Please refer to “Minimum Size and Area Rules” in LEF document to get detail information on how to check the min area rule.

Related Parameter: **check_lef_pin_min_area**

pin_min_spacing

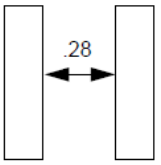
Check Item: check if the spacing between two pins, or pin and OBS, violates the min spacing constraint.

```
LAYER METAL1
    TYPE ROUTING ;
    SPACING 0.28 ;                          #Default minimum spacing is >=0.28 um
    SPACING 0.28 LENGTHTHRESHOLD 1.0 ;   #For short parallel lengths of <= 1.0 um, 0.28 spacing is allowed
    SPACING 0.32 RANGE 1.5 9.99 USELENGTHTHRESHOLD ;
                                            #Wide wires with 1.5 <= width <=9.99 need 0.32 spacing unless the
                                            #parallel run length is <= 1.0 from the previous rule
END METAL1
```

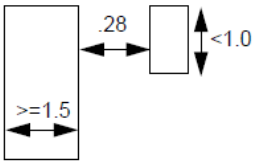
The min spacing rule is defined by SPACING, LENGTHTHRESHOLD, RANGE, USELENGTHTHRESHOLD keywords.

The spacing between two pins or pin and OBS is checked if less than the min spacing specified in the tech LEF. If the spacing is a table, only the minimum spacing will be taken because it is hard to pick up its length and width for pin and OBS to look up table.

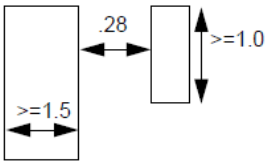
Please refer to “SPACING Statement” of routing layer in LEF document to get detail information on how to check the spacing rule.



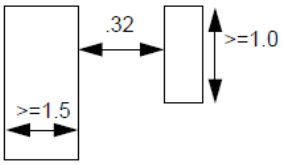
a.) No violation.



b.) No violation.



c.) Violation.



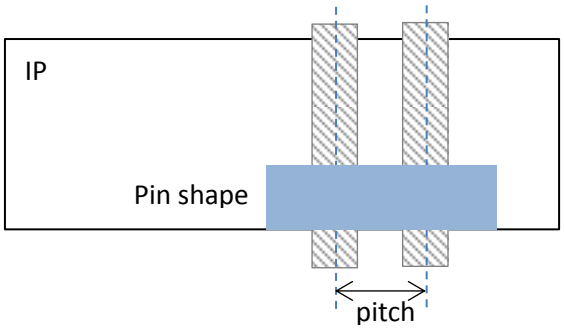
d.) No violation.

Related Parameter: **check_lef_pin_min_spacing**

pin_touch_boundary

Check Item: check if signal pin in block cell doesn't touch PR boundary (pin width touching the PR boundary should be larger than 1.5*pitch).

```
LAYER M1
  TYPE    ROUTING ;
  DIRECTION HORIZONTAL ;
  PITCH   0.14 ;
  WIDTH   0.07 ;
  MINWIDTH 0.07 ;
  MAXWIDTH 4.5 ;
  MINSTEP 0.07 MAXEDGES 1 ;
  ...
END M1
```



It will check if the signal pin does not touch the boundary or the pin width which touches the PR boundary is less than 1.5*pitch, so that routing wire can connect the shape successfully.

Power/Ground pin shapes will NOT be checked, which are specified by the parameters *power_pin_name_patterns* , and *ground_pin_name_patterns*.

This check is not the general pin width checking with min-width defined in tech LEF. It is for the block or pad design to improve quality of later routing phase. The block and pad cells are specified by the parameters *block_cell_name_patterns* , and *pad_cell_name_patterns*.

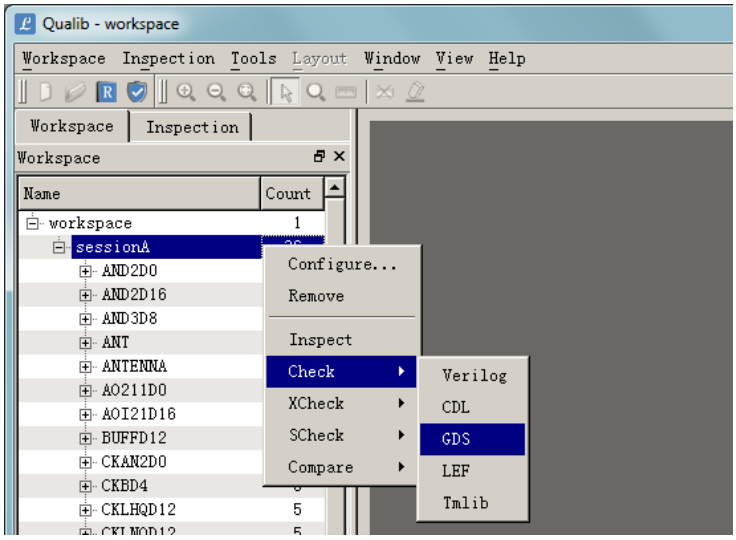
If the parameter *lef_enable_pin_width_check* is turned off, the pin width check will be disabled and only pin touching boundary will be checked.

Related Parameter: **check_lef_pin_touch_boundary**
block_cell_name_patterns
pad_cell_name_patterns
lef_enable_pin_width_check
power_pin_name_patterns
ground_pin_name_patterns

GDS

To check the GDS library files, click the GUI menu ***Inspection → Check → GDS***.

Or just right click the session name, and choose ***Check → GDS***.

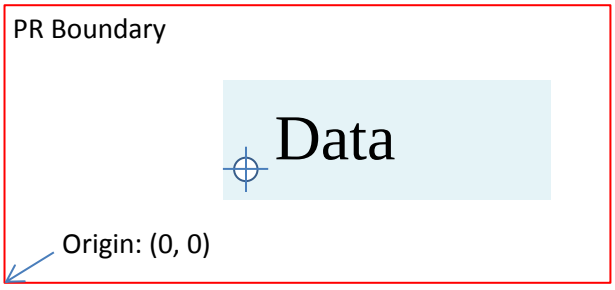



```
qualib > check -view gds  -session  sessionName
```

According to the control parameter setting of "GDS Check", enabled checking items will be performed.

boundary_origin

Check Item: check if the GDS boundary origin is not at (0, 0).

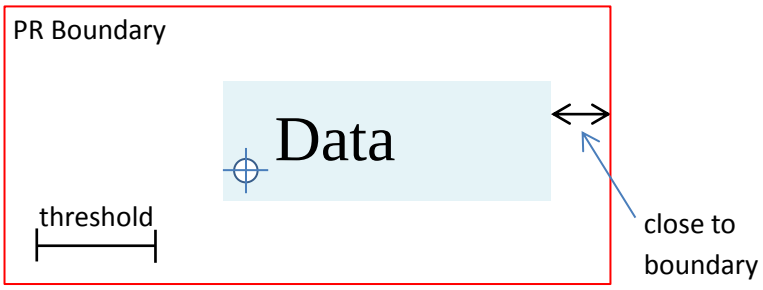


The GDS boundary layer is specified by the parameter *gds_boundary_layer.*

Related Parameter: **check_gds_boundary_origin**
gds_boundary_layer

close_to_boundary

Check Item: check if any metal shape is too close to the PR boundary.



If the distance between a shape and the PR boundary is less than the specified threshold, the shape will be reported. The distance threshold is half of min spacing defined in tech file. Power/Ground pin shapes will NOT be checked. Signal pin shapes of BLOCK and PAD cells will NOT be checked. The BLOCK and PAD cells are specified by the parameters *block_cell_name_patterns* , and *pad_cell_name_patterns*.

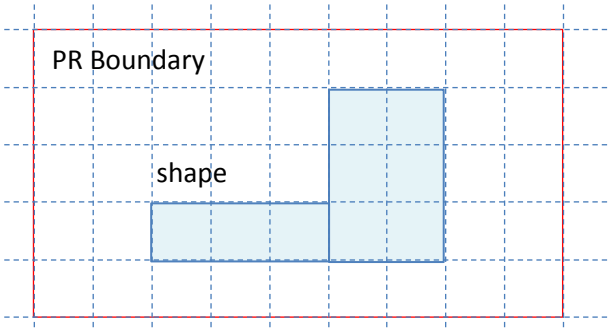
Also, the shapes on dont-check layers will be skipped. The dont check layers are specified by the parameter *gds_close_to_boundary_dont_check_layers*. The boundary layer is specified by the parameter *gds_boundary_layer*. If there are more than one boundary shapes on boundary layer, "Invalid cell boundary" will be reported and no more close to boundary checking will be done.

If parameter *gds_trace_for_pg_pin* is turned on, geometries on different layers of power/ground pins will be connected by tracing. If parameter *gds_trace_for_signal_pin* is turned on, then geometries on different layers of signal pins will be connected by tracing. The max depth for tracing is specified by parameter *gds_geometry_search_depth*. The layer connection rule is specified by tech file.

Related Parameter: **check_gds_close_to_boundary**
block_cell_name_patterns
pad_cell_name_patterns
gds_close_to_boundary_dont_check_layers
gds_boundary_layer
gds_trace_for_pg_pin
gds_trace_for_signal_pin
gds_geometry_search_depth
power_pin_name_patterns
ground_pin_name_patterns

grid_alignment

Check Item: check if any shape on metal layer is not on the manufacturing grid.



The manufacturing grid is specified by the tech file.

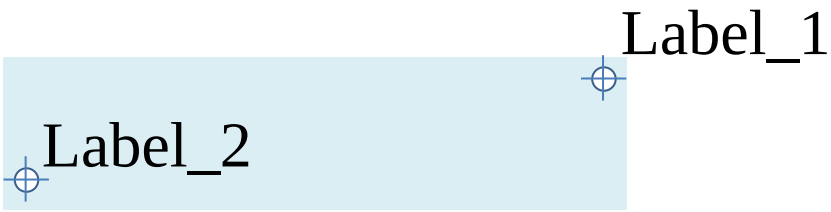
Manufacturing grid is used for geometry alignment, any shape on metal layers will be placed at locations that snap to the manufacturing grid.

For example, if the MANUFACTURINGGRID is 0.005, the shape coordinates MUST be integer multiples of 0.005, otherwise the shape is off grid.

Related Parameter: **check_gds_grid_alignment**

label_conflict

Check Item: check if there are different labels on the same shape.



If there are different labels on a shape, the shape and the labels will be reported.

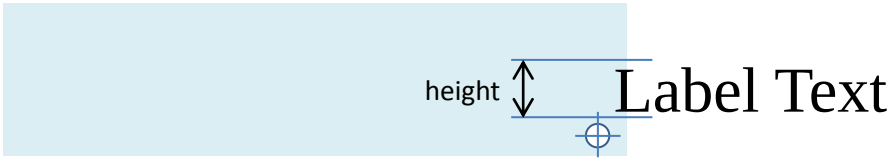
The max trace depth is specified by parameter *gds_geometry_search_depth*.

If the label layer is different with the pin layer, please specify the pin label layer map by the parameter *gds_label_layer_map*.

Related Parameter: **check_gds_label_conflict**
gds_geometry_search_depth
gds_label_layer_map

label_geometry

Check Item: check if the label height is 0.0.



If any pin label height is 0, the label will be reported.

If the label layer is different with the pin layer, please specify the pin label layer map by the parameter *gds_label_layer_map*.

Related Parameter: **check_gds_label_geometry**
gds_label_layer_map

label_layers

Check Item: check if any pin label is not on the user specified layers.

It will check if any pin label is not on the layers specified by the parameter *gds_signal_pin_label_layers* or *gds_pg_pin_label_layers*. If a pin label layer is not on the specified layers, the label will be reported.

The power pins are defined by the parameter *power_pin_name_patterns*.
The ground pins are defined by the parameter *ground_pin_name_patterns*.

If the label layer is different with the pin layer, please specify the pin label layer map by the parameter *gds_label_layer_map*.

Related Parameter: **check_gds_label_layers**
gds_signal_pin_label_layers
gds_pg_pin_label_layers
power_pin_name_patterns
ground_pin_name_patterns
gds_label_layer_map

label_shape_overlap

Check Item: check if any label has no corresponding shape.

If there is some label having not any shape, the label will be reported.

The max trace depth is determined by the parameter *gds_geometry_search_depth*.

If the label layer is different with the pin layer, please specify the pin label layer map by the parameter *gds_label_layer_map*.

Related Parameter: **check_gds_label_shape_overlap**
gds_geometry_search_depth
gds_label_layer_map

layer_hierarchy

Check Item: check if any specified layer is not on the specified hierarchy, especially the boundary layer must be on the top level.

The layer hierarchy is specified by the parameter *gds_layer_hierarchy*.
For example, "(0 (M1 drawing) (M1 text))" means on top hierarchical level, layer (M1 drawing) and layer (M1 text) should exist. Otherwise, the layer will be reported for missing layers.

If the level number is not specified, it will check the layer existence in the entire design.

The boundary layer, specified by the parameter *gds_boundary_layer*, should be on the top level.

Related Parameter: **check_gds_layer_hierarchy**
gds_layer_hierarchy
gds_boundary_layer

layer_pin_existence

Check Item: check if signal/PG pin shapes are not defined on the correct layers.

This will check if signal and P/G pin shapes exist on the layers specified by the

parameters: *gds_signal_pin_layers* and *gds_pg_pin_layers*. For the pins on some layer, if there are some missing pins not defined on some layer, the layer will be reported.

The power pin names will be defined by the parameter *power_pin_name_patterns*. The ground pin names will be defined by the parameter *ground_pin_name_patterns*.

If parameter *gds_trace_for_pg/signal_pin* is turned on, geometries on different layers of PG/signal pins will be connected by tracing. The max trace depth is specified by parameter *gds_geometry_search_depth*. The layer connection rule is specified by the tech file.

If the label layer is different with the pin layer, please specify the pin label layer map by the parameter *gds_label_layer_map*.

Related Parameter: **check_gds_layer_pin_existence**
power_pin_name_patterns
ground_pin_name_patterns
gds_signal_pin_label_layers
gds_pg_pin_label_layers
gds_trace_for_pg_pin
gds_trace_for_signal_pin
gds_geometry_search_depth
gds_label_layer_map

lib_cell_abutment

Check Item: check if any core cell violates the abutment rule.

The check is intended to ensure that the core cell can be placed adjacent to each other in rows or columns, fitting to their neighbors. The core cells are specified by the parameters *core_cell_name_patterns*, The BLOCK and PAD cells are specified by the parameters *block_cell_name_patterns*, and *pad_cell_name_patterns*.

The reference cell for GDS lib cell abutment rule check is specified by the parameter *gds_lib_abutment_reference_cell*. The layers which need to check are specified by the parameter *gds_lib_abutment_layers*. The boundary layer is specified by the parameter *gds_boundary_layer*.

For example, the power/ground lines of a cell should have the same width with the reference and occur at the same position to ensure that enough current can be provided. Otherwise, it could cause the LVS/DRC problems.

Related Parameter: **check_gds_lib_cell_abutment**
gds_lib_abutment_reference_cell
gds_lib_abutment_layers
gds_boundary_layer

block_cell_name_patterns
core_cell_name_patterns
pad_cell_name_patterns

lib_conflicting_cells

Check Item: check if any cell used in a corner is not unique.

If there are cells which have same name but their layouts are different, these cells will be reported as conflicting cells.

In the results, the same conflicting result will not be reported repeatedly. For example, if conflicting for *Cell1* between *File1* and *File2* is reported, conflicting between *File2* and *File1* will not be reported later.

Related Parameter: **check_gds_lib_conflicting_cells**

lib_database_unit

Check Item: check if the database unit conflict in gds files in a corner.

If the database unit conflict in gds files in a corner, it will be reported.

Related Parameter: **check_gds_lib_database_unit**

lib_layer_map_integrity

Check Item: check if the gds file related layers have been defined in layer map file completely.

If the layer map file does not contain all of the gds file related layers, it will be reported.

Related Parameter: **check_gds_lib_layer_map_integrity**

lib_missing_leaf_cells

Check Item: check if any leaf cell in cell list doesn't appear in any GDS file in a corner.

The leaf cell list file is specified by the parameter *gds_lib_leaf_cell_file*.

Related Parameter: **check_gds_lib_missing_leaf_cells**
gds_lib_leaf_cell_file

multi_boundary

Check Item: check if there are more than one boundaries on the top level.

The GDS boundary layer is specified by the parameter *gds_boundary_layer*.

Related Parameter: **check_gds_multi_boundary**
gds_boundary_layer

multi_ground_substrate_layer

Check Item: check if there is no extra substrate layer defined for the design with multiple GROUND layers.

If there are more than one GROUND layers in the design, while there is no extra substrate layer (specified by the parameter *gds_multi_ground_substrate_layer*), the cell will be reported.

Related Parameter: **check_gds_multi_ground_substrate_layer**
gds_multi_ground_substrate_layer

off_boundary

Check Item: check if any shape or label is out of the PR boundary.

The shape or label on dont-check layers will be skipped. The dont-check layers are specified by the parameter *gds_off_boundary_dont_check_layers*. The boundary layer is specified by the parameter *gds_boundary_layer*. If there are more than one boundary shapes on boundary layer, "Invalid cell boundary" will be reported and no more off boundary checking will be done.

In general, the power/ground pins of a standard cell are out of the boundary. So power/ground pins will only be checked for IP and PAD cells, which are specified by parameters *block_cell_name_patterns* and *pad_cell_name_patterns*.

The power pins are defined by the parameter *power_pin_name_patterns*.
The ground pins are defined by the parameter *ground_pin_name_patterns*.

If parameter *gds_trace_for_pg/signal_pin* is turned on, geometries on different layers of PG/signal pins will be connected by tracing. The max trace depth is specified by parameter *gds_geometry_search_depth*. The layer connection rule is specified by the tech file.

Related Parameter: **check_gds_off_boundary**
gds_boundary_layer
gds_off_boundary_dont_check_layers
block_cell_name_patterns
pad_cell_name_patterns
power_pin_name_patterns
ground_pin_name_patterns
gds_geometry_search_depth
gds_trace_for_pg_pin
gds_trace_for_signal_pin

pin_layers

Check Item: check if the pins are not on the user specified layers.

It will check if any pin is not on the layers specified by the parameter *gds_signal_pin_layers* or *gds_pg_pin_layers*.
If any pin is not on the specified layers, the pin will be reported.

The power pins are defined by the parameter *power_pin_name_patterns*.
The ground pins are defined by the parameter *ground_pin_name_patterns*.

If parameter *gds_trace_for_pg/signal_pin* is turned on, geometries on different layers of PG/signal pins will be connected by tracing. The max trace depth is specified by parameter *gds_geometry_search_depth*. The layer connection rule is specified by the tech file.

If the label layer is different with the pin layer, please specify the pin label layer map by the parameter *gds_label_layer_map*.

Related Parameter: **check_gds_pin_layers**
gds_pg_pin_layers
gds_signal_pin_layers
power_pin_name_patterns
ground_pin_name_patterns
gds_geometry_search_depth
gds_trace_for_pg_pin
gds_trace_for_signal_pin
gds_label_layer_map

tag_content

Check Item: check if any tag content is not matched with the user specified content.

If any tag content is not matched with the user specified content, the tag will be reported.

The tag layers are specified by the parameter *gds_tag_layers*.
The tag content is specified by the parameter *gds_tag_content*.

Related Parameter: **check_gds_tag_content**
gds_tag_layers
gds_tag_content

tag_origin

Check Item: check if the tag origin is not at the specified location.

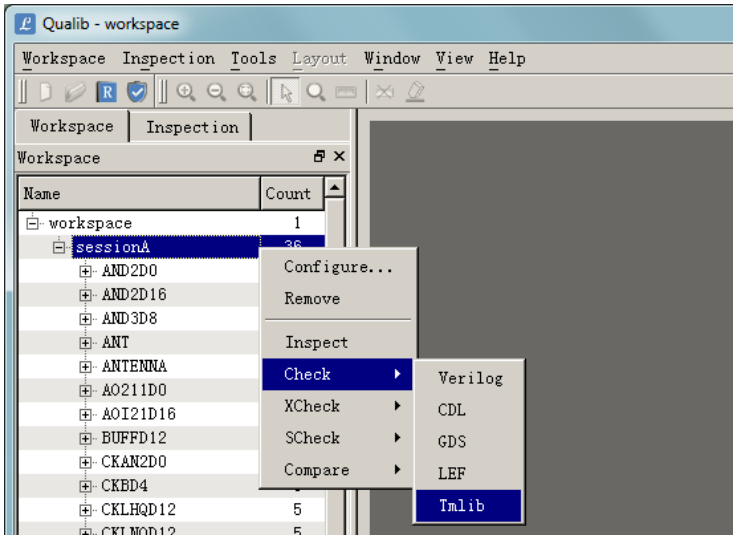
The tag origin is specified by the parameter *gds_tag_origin*.
The tag layers are specified by the parameter *gds_tag_layers*.

Related Parameter: **check_gds_tag_origin**
gds_tag_origin

Timing Lib

To check the Timing library files, click the GUI menu ***Inspection → Check → Tmlib***.

Or just right click the session name, and choose ***Check → Tmlib***.




```
qualib > check -view tmlib  -session  sessionName
```

According to the control parameter setting of "Timing Lib Check", enabled checking items will be performed.

area

Check Item: check if the cell area is undefined or redefined.

```
cell (AND2D16) {
  area : 7.7616;
  cell_footprint : "an2d1";
  ...
}
```

In the timing liberty file, if the area attribute of cell is undefined or redefined, the cell will be reported.

Related Parameter: **check_tmlib_area**

bus

Check Item: check if the bus definition (bit_from/bit_to/bit_width) is not correct.

There are five rules for bus definition check:

1) If lowest bit is equals to *tmlib_lowest_bus_bit*;
2) If *tmlib_bus_downto* is set to true, 'from' must bigger than 'to';
   If *tmlib_bus_downto* is set to false, 'to' must bigger than 'from';
3) If 'width' is equals to abs|'from' - 'to'| + 1;
4) 'downto' should be equal to the value setting in *tmlib_bus_downto*;
5) If bus bit defined as pin is not out of bus range.

If any rules are violated, the error will be reported.

Related Parameter: **check_tmlib_bus**
tmlib_bus_downto
tmlib_lowest_but_bit

ccs_index

Check Item: check if indexes of "output_current_rise()" or "output_current_fall()" group are illegal.

In "output_current_rise()" or "output_current_fall()" groups of ccs timing table, every input transition of vectors should have same list of load definition.

Also, time index should be positive and monotone increasing.

Related Parameter: **check_tmlib_ccs_index**

ccs_nldm_timing_correlation

Check Item: check if ccs and nldm delay and transition values are not correlative (relative difference larger than tolerance).

When "output_current_rise()" and "cell_rise()" are both defined in ccs and nldm model, the FO4 rise delay value will be calculated for nldm model and also for ccs model. If the relative difference is larger than the threshold specified by parameter *tmlib_delay_relative_tolerance*, it will be reported.

When "output_current_fall()" and "cell_fall()"are both defined in ccs and nldm model, the FO4 fall delay value will be calculated for nldm modle and also for ccs model. If the relative difference is larger than the threshold specified by parameter *tmlib_delay_relative_tolerance*, it will be reported.

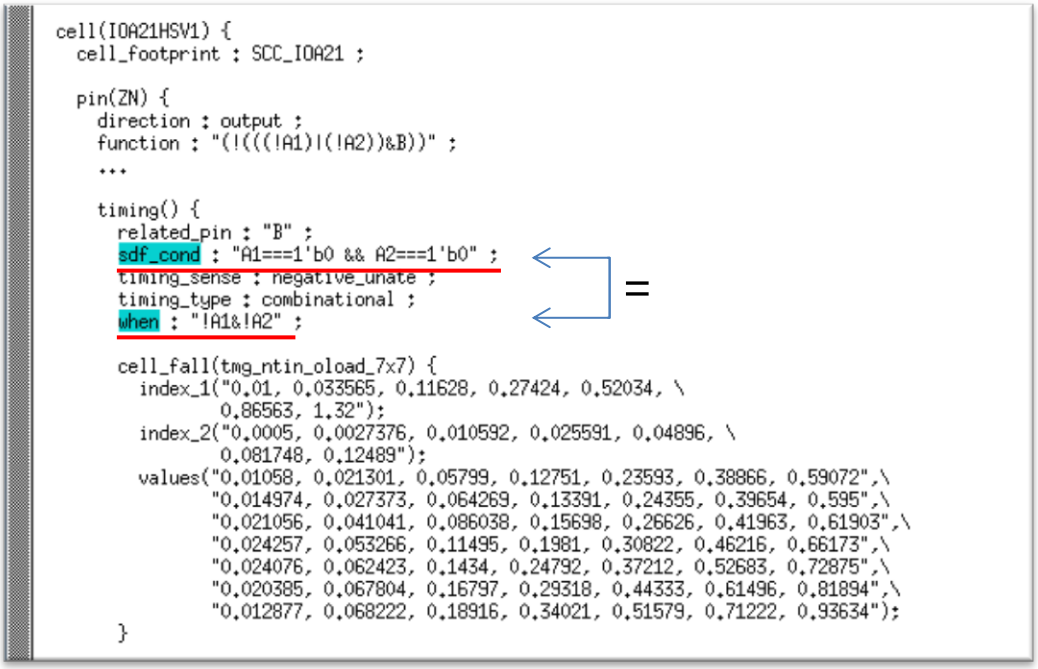
When “output_current_rise()” and “rise_transition()”are both defined in ccs and nldm model, the FO4 rise transition value will be calculated for nldm model and also for ccs model. If the relative difference is larger than the threshold specified by parameter *tmlib_transition_relative_tolerance*, it will be reported.

When “output_current_fall()” and “fall_transition()”are both defined in ccs and nldm model, the FO4 fall transition value will be calculated for nldm model and also for ccs model. If the relative difference is larger than the threshold specified by parameter *tmlib_transition_relative_tolerance*, it will be reported.

Related Parameter: **check_tmlib_ccs_nldm_timing_correlation**
tmlib_delay_relative_tolerance
tmlib_transition_relative_tolerance

condition

Check Item: check if the “when” and “sdf_cond” are not matched in a “timing()” group.



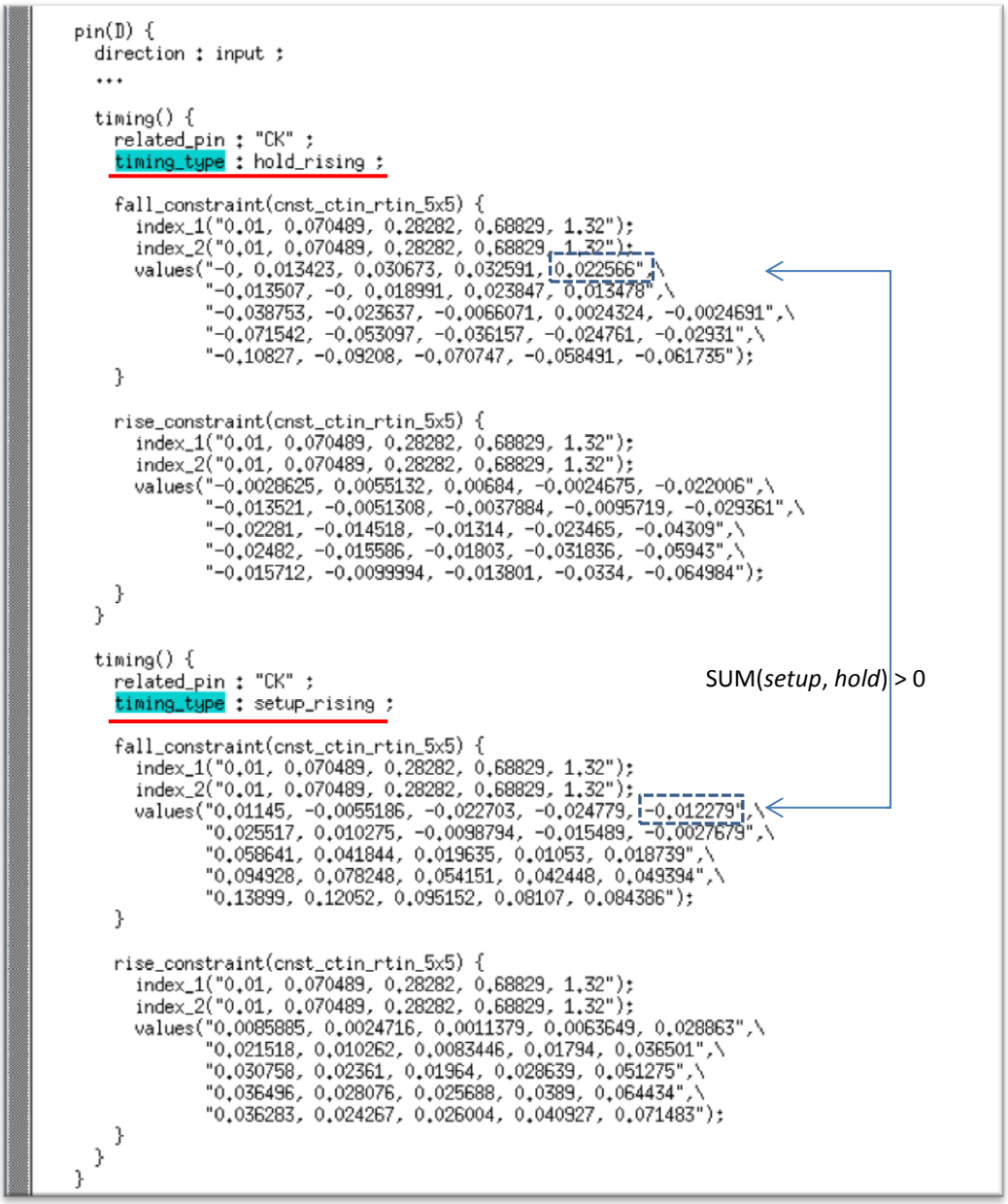
Tools will only check this item on timing groups, which the timing type is not {setup/ hold/ recovery/ removal/ min_pulse_width}.

If only “when” or “sdf_cond” attribute is defined in a “timing()” group, or functionally they are not matched, the “when” and “sdf_cond” will be reported.

Related Parameter: **check_tmlib_condition**

constraint_conflict

Check Item: check if the setup and hold, recovery and removal constraint definitions have conflict.



Setup and hold, recovery and removal constraint groups should be defined in pairs, for example, {setup_rising and hold_rising} and {setup_falling and hold_falling}. In the corresponding rise_constraint and fall_constraint timing table, the index values should be the same and the sum of each setup and hold constraint value should be positive.

If setup and hold, recovery and removal constraints have any conflict, the conflicted group and value will be reported.

Related Parameter: **check_tmlib_constraint_conflict**

default_operating_condition

Check Item: check if the default operating condition in the library is not as the user specification.

It will check if the operating condition (process, voltage, temperature and tree_type) in the library is not as user defined when creating the corner.

Related Parameter: **check_tmlib_default_operating_condition**

default_same_as_nominal

Check Item: check if the default operating condition in the library is not the same as nominal operating conditions.

It will check if the operating conditions (process, voltage, temperature) in the library is not the same as nominal operating conditions defined by nom_process, nom_voltage and nom_temperature in lib file.

Related Parameter: **check_tmlib_default_same_as_nominal**

file_name

Check Item: check if the file name is not matched with user specified format.

When user creating corners, keywords, process, voltage and temperature can be specified to those corners. If the operating conditions PVT and keyword extracted from file name are not as the user specified, then the mismatched keyword, process, voltage and temperature will be reported.

The file name format is specified in the parameter *tmlib_file_name_format* (regular expression supported):

- Process
  Process is a float value, and is defined as %P.
- Voltage
  Voltage is a float value, and is defined as %V.
- Temperature
  Temperature is a float value, and is defined as %T.
- Keyword
  Besides PVT, user often uses a nominal keyword to represent the working condition, like "bc", "wc", or "typical". This keyword can only compose of word characters: letters, numbers and underscore, and is defined as %K.

For example, *tmlib_file_name_format* is set to { _%K_%P_%Tc_%Vv }, PVT number will be extracted from file name "std_bc_1p0_m10c_1p25v.lib".

Related Parameter: **check_tmlib_file_name**
tmlib_file_name_format

footprint

Check Item: check if the cell footprint is undefined or redefined.

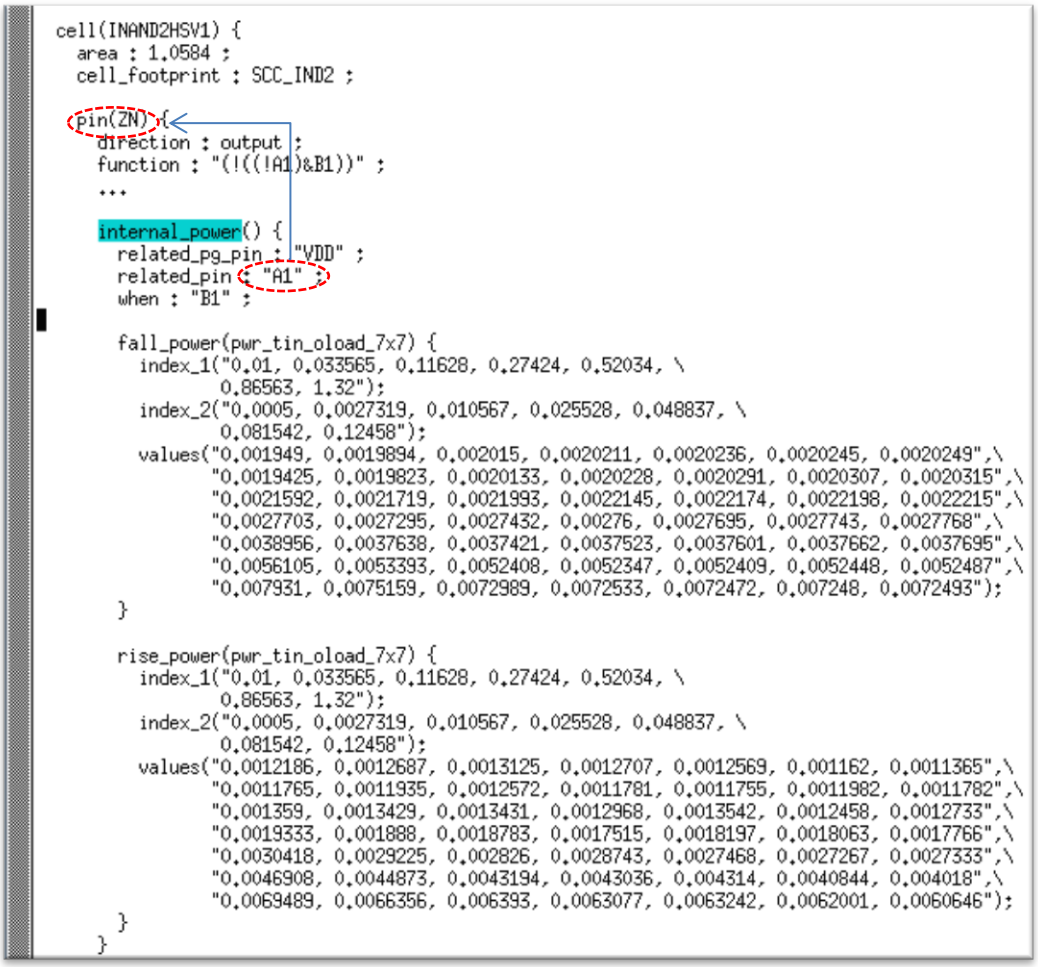
```
cell (AND2D16) {
  area : 7.7616;
  cell_footprint : "an2d1";
  ...
}
```

In the timing liberty file, if the cell_footprint attribute of cell is undefined or redefined, the cell will be reported.

Related Parameter: **check_tmlib_footprint**

internal_power_arc_presence

Check Item: check if the internal power arcs are not completely defined.



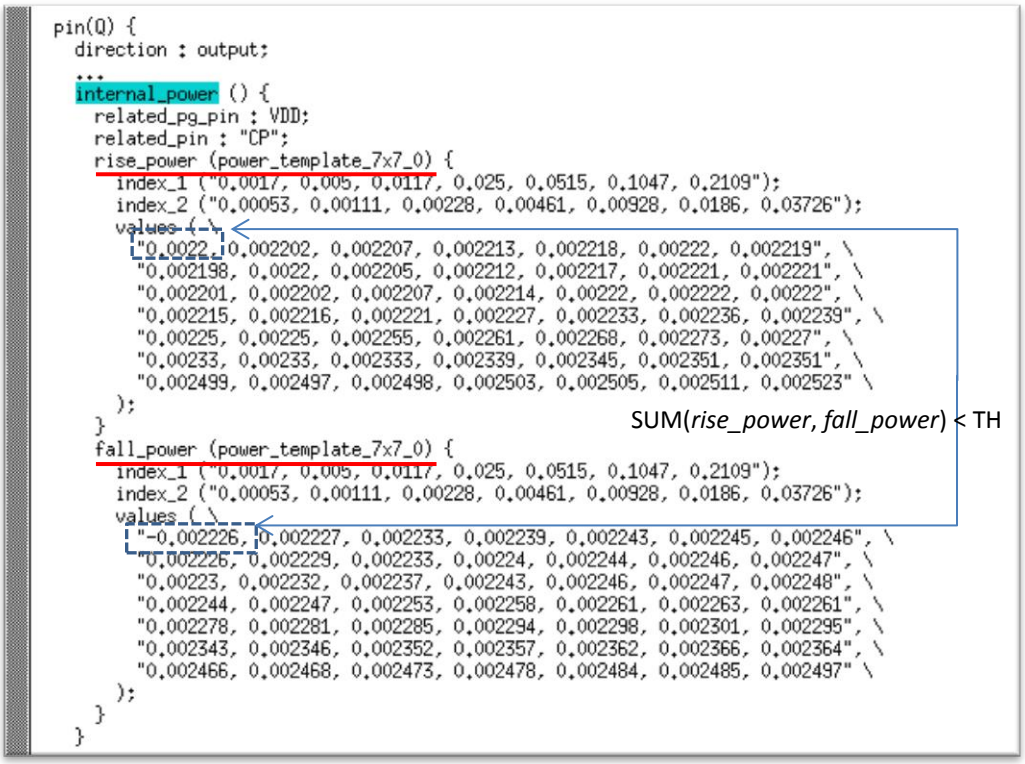
```
cell(INAND2HSV1) {
  area : 1.0584 ;
  cell_footprint : SCC_IND2 ;

  pin(ZN) {
    direction : output ;
    function : "(!((!A1)&B1))" ;
    ...

    internal_power() {
      related_pg_pin : "VDD" ;
      related_pin : "A1" ;
      when : "B1" ;

      fall_power(pwr_tin_oload_7x7) {
        index_1("0.01, 0.033565, 0.11628, 0.27424, 0.52034, \
                 0.86563, 1.32");
        index_2("0.0005, 0.0027319, 0.010567, 0.025528, 0.048837, \
                 0.081542, 0.12458");
        values("0.001949, 0.0019894, 0.002015, 0.0020211, 0.0020236, 0.0020245, 0.0020249",\
               "0.0019425, 0.0019823, 0.0020133, 0.0020228, 0.0020291, 0.0020307, 0.0020315",\
               "0.0021592, 0.0021719, 0.0021993, 0.0022145, 0.0022174, 0.0022198, 0.0022215",\
               "0.0027703, 0.0027295, 0.0027432, 0.00276, 0.0027695, 0.0027743, 0.0027768",\
               "0.0038956, 0.0037638, 0.0037421, 0.0037523, 0.0037601, 0.0037662, 0.0037695",\
               "0.0056105, 0.0053393, 0.0052408, 0.0052347, 0.0052409, 0.0052448, 0.0052487",\
               "0.007931, 0.0075159, 0.0072989, 0.0072533, 0.0072472, 0.007248, 0.0072493");
      }

      rise_power(pwr_tin_oload_7x7) {
        index_1("0.01, 0.033565, 0.11628, 0.27424, 0.52034, \
                 0.86563, 1.32");
        index_2("0.0005, 0.0027319, 0.010567, 0.025528, 0.048837, \
                 0.081542, 0.12458");
        values("0.0012186, 0.0012687, 0.0013125, 0.0012707, 0.0012569, 0.001162, 0.0011365",\
               "0.0011765, 0.0011935, 0.0012572, 0.0011781, 0.0011755, 0.0011982, 0.0011782",\
               "0.001359, 0.0013429, 0.0013431, 0.0012968, 0.0013542, 0.0012458, 0.0012733",\
               "0.0019333, 0.001888, 0.0018783, 0.0017515, 0.0018197, 0.0018063, 0.0017766",\
               "0.0030418, 0.0029225, 0.002826, 0.0028743, 0.0027468, 0.0027267, 0.0027333",\
               "0.0046908, 0.0044873, 0.0043194, 0.0043036, 0.004314, 0.0040844, 0.004018",\
               "0.0069489, 0.0066356, 0.006393, 0.0063077, 0.0063242, 0.0062001, 0.0060646");
      }
    }
  }
```

For combinational cells, internal power arc should be from input to output. Input must appear on output's 'function'. For sequential cells, internal power will be checked on input pins (except power/ground pins).

Related Parameter: **check_tmlib_internal_power_arc_presence**

internal_power_conflict

Check Item: check if the sum of rise and fall power value is less than the user specified threshold in “internal_power()” group.

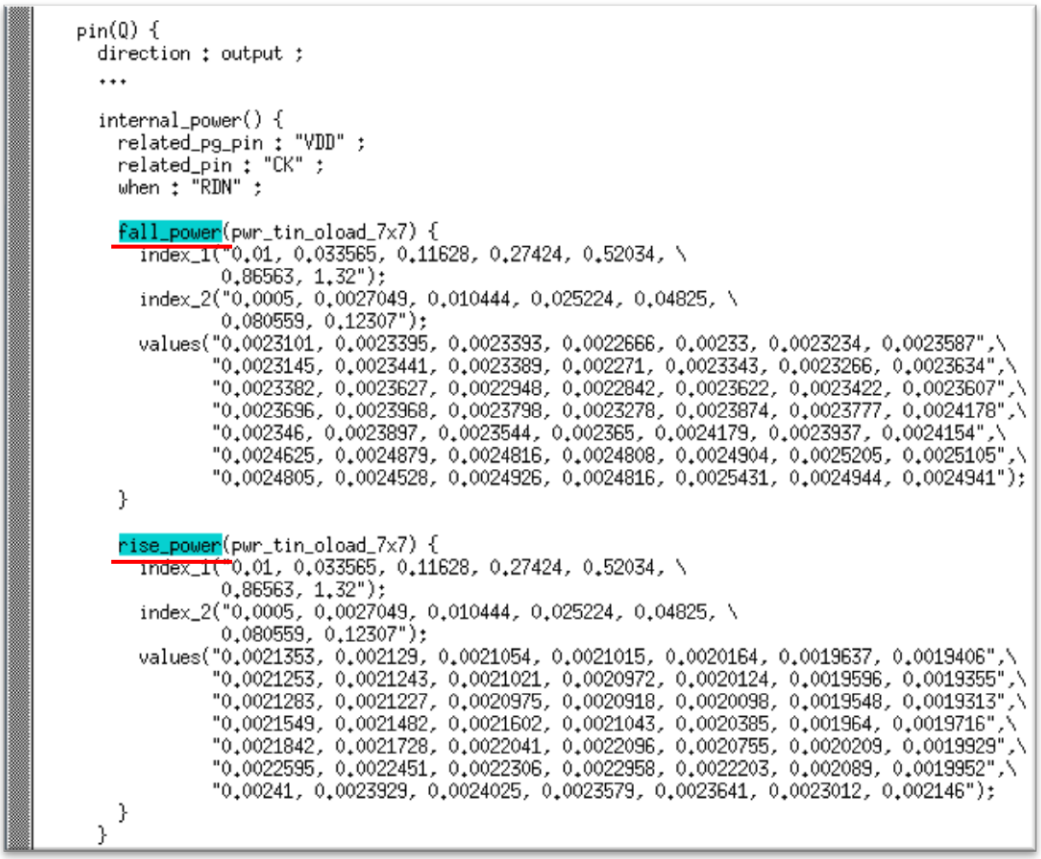


In 'internal_power()' group in a timing library file, if the sum of 'rise_power' and 'fall_power' is less than the threshold specified by the parameter *tmlib_internal_power_conflict_threshold*, the conflict value will be reported.

Related Parameter: **check_tmlib_internal_power_conflict**
tmlib_internal_power_conflict_threshold

internal_power_table_presence

Check Item: check if the internal power tables are not completely defined.



```
pin(Q) {
  direction : output ;
  ...

  internal_power() {
    related_pg_pin : "VDD" ;
    related_pin : "CK" ;
    when : "RDN" ;

    fall_power(pwr_tin_oload_7x7) {
      index_1("0.01, 0.033565, 0.11628, 0.27424, 0.52034, \
               0.86563, 1.32");
      index_2("0.0005, 0.0027049, 0.010444, 0.025224, 0.04825, \
               0.080559, 0.12307");
      values("0.0023101, 0.0023395, 0.0023393, 0.0022666, 0.00233, 0.0023234, 0.0023587",\
             "0.0023145, 0.0023441, 0.0023389, 0.002271, 0.0023343, 0.0023266, 0.0023634",\
             "0.0023382, 0.0023627, 0.0022948, 0.0022842, 0.0023622, 0.0023422, 0.0023607",\
             "0.0023696, 0.0023968, 0.0023798, 0.0023278, 0.0023874, 0.0023777, 0.0024178",\
             "0.002346, 0.0023897, 0.0023544, 0.002365, 0.0024179, 0.0023937, 0.0024154",\
             "0.0024625, 0.0024879, 0.0024816, 0.0024808, 0.0024904, 0.0025205, 0.0025105",\
             "0.0024805, 0.0024528, 0.0024926, 0.0024816, 0.0025431, 0.0024944, 0.0024941");
    }

    rise_power(pwr_tin_oload_7x7) {
      index_1("0.01, 0.033565, 0.11628, 0.27424, 0.52034, \
               0.86563, 1.32");
      index_2("0.0005, 0.0027049, 0.010444, 0.025224, 0.04825, \
               0.080559, 0.12307");
      values("0.0021353, 0.002129, 0.0021054, 0.0021015, 0.0020164, 0.0019637, 0.0019406",\
             "0.0021253, 0.0021243, 0.0021021, 0.0020972, 0.0020124, 0.0019596, 0.0019355",\
             "0.0021283, 0.0021227, 0.0020975, 0.0020918, 0.0020098, 0.0019548, 0.0019313",\
             "0.0021549, 0.0021482, 0.0021602, 0.0021043, 0.0020385, 0.001964, 0.0019716",\
             "0.0021842, 0.0021728, 0.0022041, 0.0022096, 0.0020755, 0.0020209, 0.0019929",\
             "0.0022595, 0.0022451, 0.0022306, 0.0022958, 0.0022203, 0.002089, 0.0019952",\
             "0.00241, 0.0023929, 0.0024025, 0.0023579, 0.0023641, 0.0023012, 0.002146");
    }
  }
```

Internal power group is usually defined on a power arc or on a pin.
If the *rise_power* nor *fall_power* tables are not in pair, or there is no *power* table, the cell and the pins will be reported.

Related Parameter: **check_tmlib_internal_power_table_presence**

leakage_power

Check Item: check if the cell leakage power value is not within the user specified range.

```
cell (AND2D16) {
  area : 7.7616;
  cell_footprint : "an2d1";
  ...
  leakage_power () {
    value : 45.681732;
    when : "!A1 !A2 !Z";
    related_pg_pin : VDD;
  }
```

If the cell leakage power value (defined either by cell_leakage_power or "leakage_power()" group) is not within the range defined by the parameter *tmlib_leakage_power_range*, the cell and the leakage power value will be reported.

Cells defined in parameter *tmlib_leakage_power_dont_check_cell_name_patterns* will be skipped for this check.

Related Parameter: **check_tmlib_leakage_power**
tmlib_leakage_power_range
tmlib_leakage_power_dont_check_cell_name_patterns

lib_slew_delay_definition

Check Item: check if the slew or delay definition is not same as the user specified.

```
library (ssh1) {
slew_lower_threshold_pct_rise :  30.00
slew_upper_threshold_pct_rise :  70.00
slew_lower_threshold_pct_fall :  30.00
slew_upper_threshold_pct_fall :  70.00
slew_derate_from_library      :  0.5
input_threshold_pct_rise      :  50.0;
input_threshold_pct_fall      :  50.0;
output_threshold_pct_rise     :  50.0;
output_threshold_pct_fall     :  50.0;
...
}
```

The slew definitions are specified by the parameters *tmlib_slew_lower_threshold_pct*, *tmlib_slew_upper_threshold_pct*, *tmlib_slew_derate_threshold*.

The delay definitions are specified by the parameters *tmlib_delay_input_threshold_pct*, *tmlib_delay_output_threshold_pct*.

Related Parameter: **check_tmlib_lib_slew_delay_definition**
tmlib_slew_derate_threshold
tmlib_delay_input_threshold_pct
tmlib_delay_output_threshold_pct
tmlib_slew_lower_threshold_pct
tmlib_slew_upper_threshold_pct

lib_unit_definition

Check Item: check if unit is not defined or definition in the library is not the same in corner.

It will check if time/voltage/leakage power/capacitance load/pulling resistance/current unit is defined and the definitions are same in corner.

Related Parameter: **check_tmlib_lib_unit_definition**

library_name

Check Item: check if the lib name is not matched with user specified format.

When user creating corners, keywords, process, voltage and temperature can be specified to those corners. If the operating conditions PVT and keywords extracted from lib name are not as the user specified, then the mismatched keywords, process, voltage and temperature in library name will be reported.

The lib name format is specified in parameter *tmlib_library_name_format* (regular expression supported):

- Process
  Process is a float value, and is defined as %P.
- Voltage
  Voltage is a float value, and is defined as %V.
- Temperature
  Temperature is a float value, and is defined as %T.
- Keyword
  Besides PVT, user often uses a nominal keyword to represent the working condition, like "bc", "wc", or "typical". This keyword can only compose of word characters: letters, numbers and underscore, and is defined as %K.

For example, *tmlib_library_name_format* is set to { _%K_%P_%Tc_%Vv }, PVT number will be extracted from lib name "std_bc_1p0_m10c_1p25v".

Related Parameter: **check_tmlib_library_name**
tmlib_library_name_format

library_name_in_corner

Check Item: check if library name is not the same as other library names in same corner.

Library names defined in same corner should be same. If there is any library name not the same as other library names, it will be reported.

For example, there are four .lib files in *wcl* corner. 3 of 4 library names are *tutorial0p81m40cwcl*, the other library name is *tutorial0p81m40cbc*, this different library name will be reported.

Related Parameter: **check_tmlib_library_name_in_corner**

noise_table_presence

Check Item: check if ccsn table is not completely defined in ccsn group.

Completeness of “ccsn_first_stage()” and “ccsn_last_stage()” section in the timing library file means *dc_current* and *output_voltage_fall* and *output_voltage_rise* and *propagated_noise_high* and *propagated_noise_low* tables will be checked if they are defined in the timing table.

When “stage_type” is defined to “pull up”, *output_voltage_rise*, *propagated_noise_low*, *dc_current* will be checked; when it is defined to “pull down”, *output_voltage_fall*, *propagated_noise_high*, *dc_current* will be checked; when it is not defined or defined as both, all tables will be checked. If any table is missed, the table name will be reported.

Related Parameter: **check_tmlib_noise_table_presence**

operating_conditions_name

Check Item: check if the operating condition name is not matched with user specified format.

It will check if operating conditions name is not matched with user specified format. If the format is not violated, we will check if the PVT not as the user specified when creating corners.

The operating conditions name format is specified in parameter *tmlib_operating_conditions_name_format* (regular expression supported):

- Process
  Process is a float value, and is defined as %P.
- Voltage
  Voltage is a float value, and is defined as %V.
- Temperature
  Temperature is a float value, and is defined as %T.
- Keyword
  Besides PVT, user often uses a nominal keyword to represent the working condition, like "bc", "wc", or "typical". This keyword can only compose of word characters: letters, numbers and underscore, and is defined as %K.

For example, *tmlib_operating_conditions_name_format* is {%V_%Tc_%K_%P}, PVT number will be extracted from operating_conditions " 0p81_m40c_wc_l".

Related Parameter: **check_tmlib_operating_conditions_name**
tmlib_operating_conditions_name_format

pg_pin_with_voltage_name

Check Item: check if voltage name is not defined on voltage maps.

If *'voltage_name*' attribute is not specified, undefined will be reported.

If '*voltage_name*' is not matched with '*voltage_map*' defined on library level, unmatched voltage_name will be reported.

Related Parameter: **check_tmlib_pg_pin_with_voltage_name**

pin_with_capacitance

Check Item: check if the pin capacitance is not defined for an input pin, or the value is not within user specified range.

Related Parameter: **check_tmlib_pin_with_capacitance**
tmlib_pin_capacitance_range

pin_with_max_capacitance

Check Item: check if max_capacitance is not defined for an output pin, or the max_capacitance is out of range.

If max capacitance is not defined, the pin will be reported.
If max capacitance value is not in the range defined by parameter *tmlib_max_capacitance_range*, it will report max capacitance required range and the actual value.

Cells defined in parameter *tmlib_cap_constraint_dont_check_cell_name_patterns* will be skipped for this check.

Related Parameter: **check_tmlib_pin_with_max_capacitance**
tmlib_max_capacitance_range
tmlib_cap_constraint_dont_check_cell_name_patterns

pin_with_max_transition

Check Item: check if max_transition is not defined for an input pin, or the max_transition is out of range.

```
library(lib_name) {
  delay_model : table_lookup ;
  revision : "0.2" ;
  time_unit : 1ns ;
  voltage_unit : 1V ;
  current_unit : 1mA ;
  leakage_power_unit : 1uW ;
  default_fanout_load : 1 ;
  default_inout_pin_cap : 0.0045094;
  default_input_pin_cap : 0.0045094;
  default_max_transition : 1.31 ;
  default_output_pin_cap : 0 ;
```

If the max transition is not defined, the pin will be reported.
If the max transition value is not in the range defined by parameter *tmlib_max_transition_range*, it will report the max transition required range and the actual value.

Related Parameter: **check_tmlib_pin_with_max_transition**
tmlib_max_transition_range

pin_with_min_capacitance

Check Item: check if min_capacitance is not defined for an output pin.

```
pin("paddr[8]") {
  direction : output ;
  max_transition : 0.210900 ;
  min_transition : 0.000000 ;
  max_capacitance : 0.136200 ;
  min_capacitance : 0.000000 ;
  capacitance : 0.004953 ;

  /* Other user defined attributes. */
  original_pin : paddr[8];

} /* end of pin paddr[8] */
```

The “min_capacitance” attribute is defined to constrain output pin capacitance.

Cells defined in parameter *tmlib_cap_constraint_dont_check_cell_name_patterns* will be skipped for this check.

Related Parameter: **check_tmlib_pin_with_min_capacitance**
tmlib_cap_constraint_dont_check_cell_name_patterns

pin_with_min_pulse_width

Check Item: check if min_pulse_width is not defined for a clock pin.

```
pin(CP) {
  clock : true;
  direction : input;
  ...
  timing () {
    related_pin : "CP";
    sdf_cond : "E_NTE_SDFCHK";
    timing_type : min_pulse_width;
    when : "E&TE";
    rise_constraint (mpw_constraint_template_3x3) {
      index_1 ("0.0017, 0.025, 0.2109");
      values ( \
        "0.01221, 0.03174, 0.2612" \
      );
    }
    fall_constraint (mpw_constraint_template_3x3) {
      index_1 ("0.0017, 0.025, 0.2109");
      values ( \
        "0.01221, 0.03174, 0.2612" \
      );
    }
  }
```

The “min_pulse_width” timing group is defined to constrain a clock signal.

Related Parameter: **check_tmlib_pin_with_min_pulse_width**

pin_with_power_down_function

Check Item: check if 'power_down_function' is not defined on output/inout pin.

If *tmlib_power_down_function_verification* is set to true, it will check whether 'power_down_function' is matched sum of all the !power and ground. If not matching, 'power_down_function' and reference function will be reported.

Related Parameter: **check_tmlib_pin_with_power_down_function**
tmlib_power_down_function_verification

pin_with_related_ground_pin

Check Item: check if related ground pin is not defined on pin or not defined as ground pin.

If related_ground_pin attribute is not be specified, undefined will be reported.

If related_ground_pin is not defined as pg_pin on cell level, related_ground_pin not as required will be reported.

Related Parameter: **check_tmlib_pin_with_related_ground_pin**

pin_with_related_power_pin

Check Item: check if related power pin is not defined on pin or not defined as power pin.

If related_power_pin attribute is not be specified, undefined will be reported.

If related_power_pin is not defined as pg_pin on cell level, related_power_pin not as required will be reported.

Related Parameter: **check_tmlib_pin_with_related_power_pin**

table_index_conflict

Check Item: check if the indexes for all tables of “timing()” or “internal_power()” group are not identical.

In 'timing()' groups, all of tables {*rise_transition*, *fall_transition*, *cell_rise*, *cell_fall*, *output_current_rise* and *output_current_fall*} should have the same index definition. The timing constraint definition {*rise_constraint* and *fall_constraint*} in one group should have the same index definition.

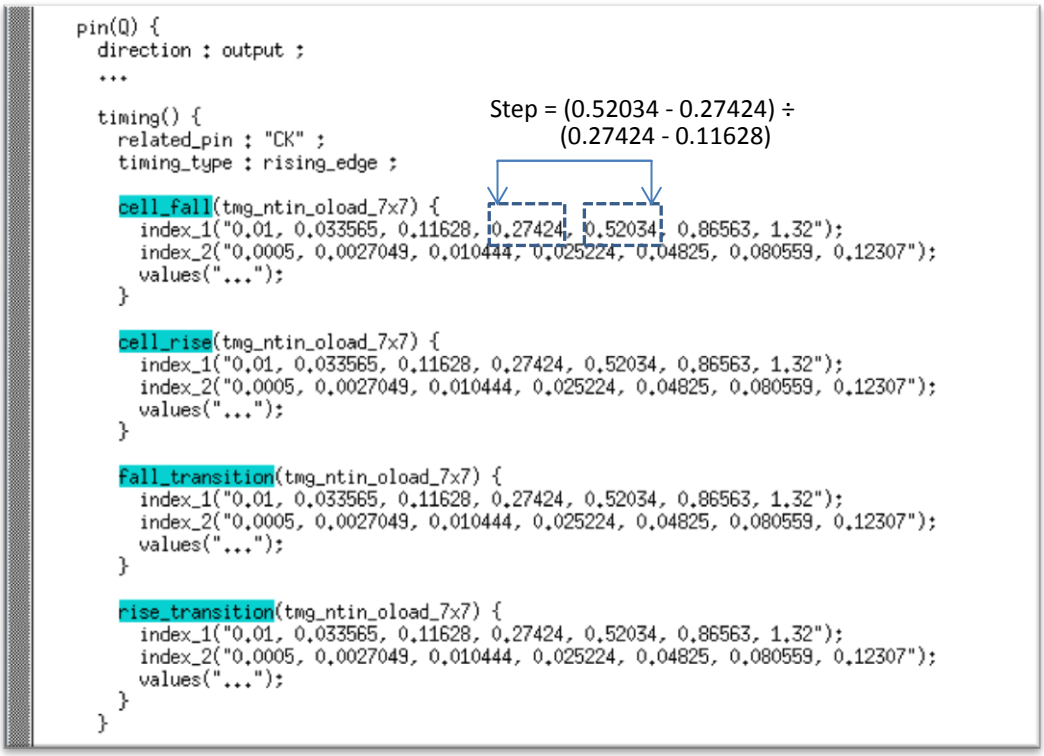
In 'internal_power()' groups, all of tables {*rise_power* and *fall_power*} should have the same index definition.

The threshold is specified in parameter *tmlib_table_index_conflict_threshold*. If any table has different index definition (index size or index content), it will be reported.

Related Parameter: **check_tmlib_table_index_conflict**
tmlib_table_index_conflict_threshold
tmlib_table_index_conflict_skip_different_dimension_tables

table_index_step_growth_rate

Check Item: Check if the index step growing rate of a table is greater than the user specified threshold, or the index for power/timing table is not monotonic.




```
pin(Q) {
  direction : output ;
  ...

  timing() {
    related_pin : "CK" ;
    timing_type : rising_edge ;

    cell_fall(tmg_ntin_oload_7x7) {
      index_1("0.01, 0.033565, 0.11628, 0.27424, 0.52034, 0.86563, 1.32");
      index_2("0.0005, 0.0027049, 0.010444, 0.025224, 0.04825, 0.080559, 0.12307");
      values("...");
    }

    cell_rise(tmg_ntin_oload_7x7) {
      index_1("0.01, 0.033565, 0.11628, 0.27424, 0.52034, 0.86563, 1.32");
      index_2("0.0005, 0.0027049, 0.010444, 0.025224, 0.04825, 0.080559, 0.12307");
      values("...");
    }

    fall_transition(tmg_ntin_oload_7x7) {
      index_1("0.01, 0.033565, 0.11628, 0.27424, 0.52034, 0.86563, 1.32");
      index_2("0.0005, 0.0027049, 0.010444, 0.025224, 0.04825, 0.080559, 0.12307");
      values("...");
    }

    rise_transition(tmg_ntin_oload_7x7) {
      index_1("0.01, 0.033565, 0.11628, 0.27424, 0.52034, 0.86563, 1.32");
      index_2("0.0005, 0.0027049, 0.010444, 0.025224, 0.04825, 0.080559, 0.12307");
      values("...");
    }
  }
```

This item will be checked on timing groups (with timing type cell_rise/ cell_fall/ rise_transition/ fall_transition) and internal_power groups.

The index step growth rate is calculated by the index margin dividing, which index margin is Xn-Xn-1.

It should not be larger than the threshold value, which is specified by parameter *tmlib_power_table_index_step_growth_rate* and *tmlib_timing_table_index_step_growth_rate*

Related Parameter: **check_tmlib_table_index_step_growth_rate**
tmlib_power_table_index_step_growth_rate
tmlib_timing_table_index_step_growth_rate

table_value

Check Item: check if the value of timing tables is not in the specified range.

The valid value ranges are specified by parameter *tmlib_constraint_value_range*, *tmlib_delay_value_range*, *tmlib_internal_power_value_range* and *tmlib_transition_value_range*. The table will be reported if its contents are beyond the specified range.

Related Parameter: **check_tmlib_table_value**
tmlib_constraint_value_range
tmlib_delay_value_range
tmlib_internal_power_value_range
tmlib_transition_value_range

timing_arc_presence

Check Item: check if the timing arcs are not completely defined.

Timing arc can be defined in not only combinational cells but also clock cells. For a combinational cell, it is from an input pin to an output pin; for a clock cell, the timing arc is from the clock pin to an output data pin.

If not all the timing arcs of cell are defined, the missing one will be reported.

Related Parameter: **check_tmlib_timing_arc_presence**

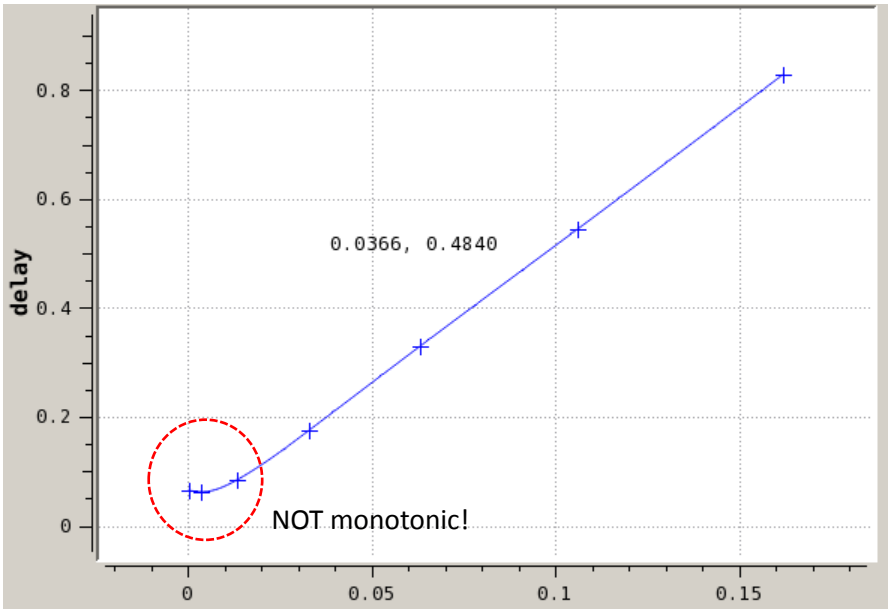
timing_monotonicity_with_load

Check Item: check if the output delay or transition is not monotonic with the output load.

```
timing() {
  related_pin : "SDN" ;
  sdf_cond : "CK===1'b0 && D===1'b0" ;
  timing_sense : positive_unate ;
  timing_type : clear ;
  when : "!CK&!D" ;

  fall_transition(tmg_ntin_oload_7x7) {
    index_1("0.01, 0.033565, 0.11628, 0.27424, 0.52034, \          <-- input transition
          0.86563, 1.32");
    index_2("0.0005, 0.0033998, 0.013579, 0.033017, 0.063301, \    <-- output capacitance
          0.10579, 0.16171");
    values("0.02144, 0.034603, 0.075325, 0.1725, 0.32847, 0.54355, 0.82958",\
           "0.021342, 0.034712, 0.075455, 0.17322, 0.32775, 0.54599, 0.82913",\
           "0.025285, 0.036311, 0.075537, 0.1732, 0.32793, 0.54606, 0.8289",\
           "0.035278, 0.043395, 0.077396, 0.1727, 0.32704, 0.54333, 0.82962",\
           "0.04548, 0.050489, 0.079546, 0.17347, 0.32704, 0.5438, 0.82848",\
           "0.055491, 0.056384, 0.081947, 0.1736, 0.32887, 0.54399, 0.82793",\
           "0.064784, 0.061567, 0.084154, 0.17391, 0.32885, 0.54565, 0.82915");
  }
}
```

This item will only be checked on timing groups. Four types of timing table {cell_rise/ cell_fall/ rise_transition/ fall_transition} will be checked.

If the output delay or transition is not monotonic (difference larger than threshold value specified by the parameter *tmlib_monotonicity_threshold*) with the output load in a "timing()" group, the table will be reported.



Related Parameter: **check_tmlib_timing_monotonicity_with_load**
tmlib_monotonicity_threshold

timing_monotonicity_with_slew

Check Item: check if the output delay or transition is not monotonic with the input transition.

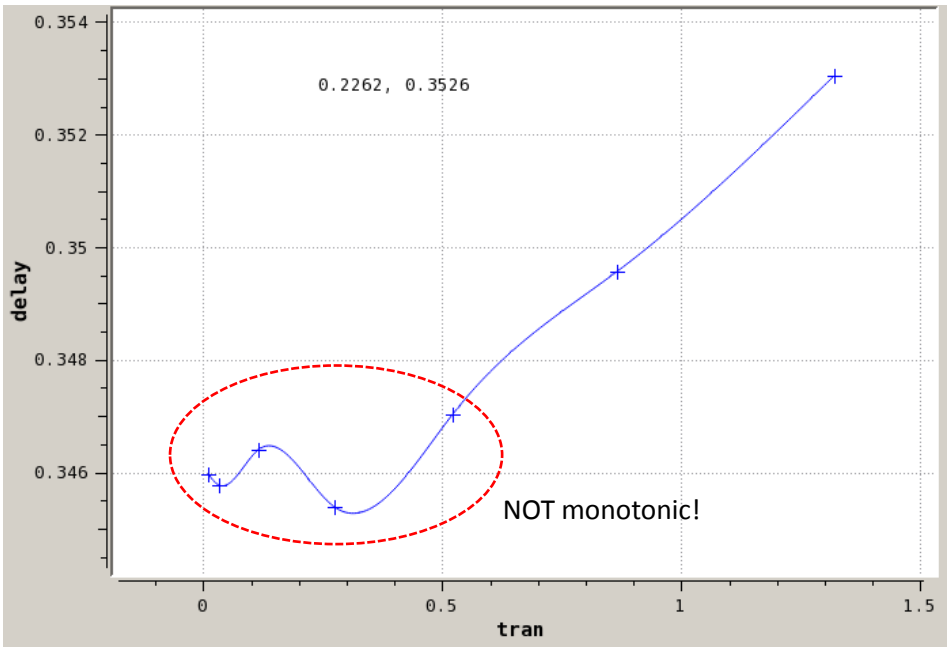
```
timing() {
  related_pin : "A2" ;
  timing_sense : positive_unate ;
  timing_type : combinational ;
  ...

  fall_transition(tmg_ntin_oload_7x7) {
    index_1("0.01, 0.033565, 0.11628, 0.27424, 0.52034, \
             0.86563, 1.32");
    index_2("0.0005, 0.013085, 0.057263, 0.14163, 0.27306, \
             0.45748, 0.70015");
    values("0.010979, 0.025033, 0.076709, 0.18198, 0.34597, 0.57534, 0.87675",\
           "0.011138, 0.02507,  0.07669,  0.18181, 0.34576, 0.57643, 0.87638",\
           "0.017539, 0.02994,  0.07765,  0.18194, 0.34641, 0.57562, 0.87849",\
           "0.026741, 0.038804, 0.081808, 0.18201, 0.34539, 0.57582, 0.87779",\
           "0.037382, 0.049583, 0.089904, 0.18436, 0.34704, 0.57431, 0.87707",\
           "0.050558, 0.062517, 0.102,    0.19008, 0.34957, 0.57664, 0.87626",\
           "0.065581, 0.077106, 0.11865,  0.19919, 0.35304, 0.57921, 0.87905");
  }
```

input transition

output capacitance

This item will only be checked on timing groups (with timing type cell_rise/ cell_fall/ rise_transition/ fall_transition).

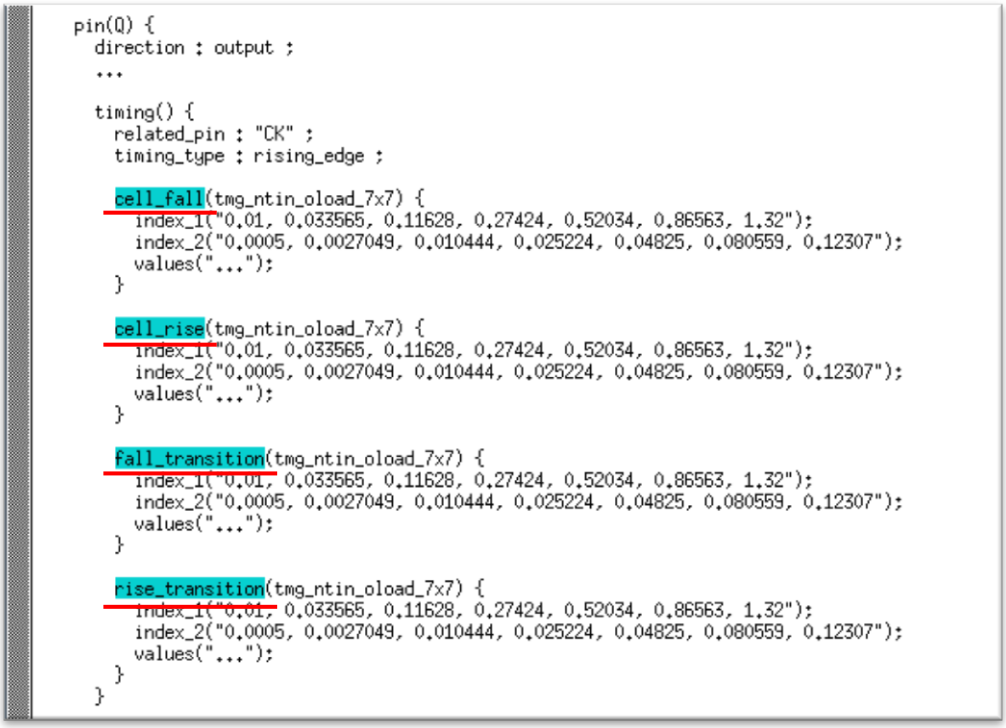
If the output delay or transition is not monotonic (difference larger than threshold value specified by the parameter *tmlib_monotonicity_threshold*) with the input slew in a “timing()” group, the table will be reported.



Related Parameter: **check_tmlib_timing_monotonicity_with_slew**
tmlib_monotonicity_threshold

timing_table_presence

Check Item: check if the timing table is not completely defined for timing() group.



```
pin(Q) {
  direction : output ;
  ...

  timing() {
    related_pin : "CK" ;
    timing_type : rising_edge ;

    cell_fall(tmg_ntin_oload_7x7) {
      index_1("0.01, 0.033565, 0.11628, 0.27424, 0.52034, 0.86563, 1.32");
      index_2("0.0005, 0.0027049, 0.010444, 0.025224, 0.04825, 0.080559, 0.12307");
      values("...");
    }

    cell_rise(tmg_ntin_oload_7x7) {
      index_1("0.01, 0.033565, 0.11628, 0.27424, 0.52034, 0.86563, 1.32");
      index_2("0.0005, 0.0027049, 0.010444, 0.025224, 0.04825, 0.080559, 0.12307");
      values("...");
    }

    fall_transition(tmg_ntin_oload_7x7) {
      index_1("0.01, 0.033565, 0.11628, 0.27424, 0.52034, 0.86563, 1.32");
      index_2("0.0005, 0.0027049, 0.010444, 0.025224, 0.04825, 0.080559, 0.12307");
      values("...");
    }

    rise_transition(tmg_ntin_oload_7x7) {
      index_1("0.01, 0.033565, 0.11628, 0.27424, 0.52034, 0.86563, 1.32");
      index_2("0.0005, 0.0027049, 0.010444, 0.025224, 0.04825, 0.080559, 0.12307");
      values("...");
    }
  }
```

Completeness of a “timing()” group in a timing library file means all of {rise_transition/ fall_transition/ cell_rise/ cell_fall} tables exist. If any one of the four tables is missing, the table will be reported.

Related Parameter: **check_tmlib_timing_table_presence**

voltage_map_definition

Check Item: check if voltage map is not defined in the library or voltage values 0 is not appeared.

The library-level '*voltage_map*' attribute associates the voltage name with the relative voltage values. These specified voltage names are referenced by the pg_pin groups defined at the cell level. At least one voltage map in the library should have a value of 0.

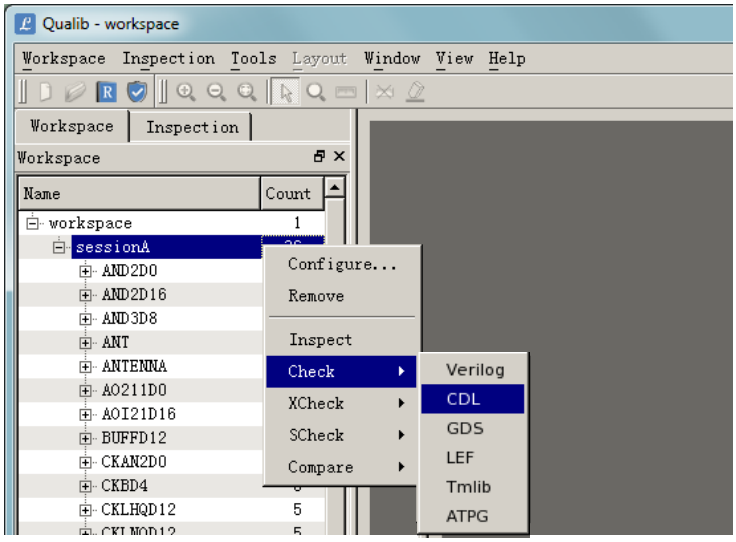
If attribute '*voltage_map*' is not defined in the power and ground pin library, voltage_map undefined will be reported. If no '*voltage_map*' values 0, no voltage_map values 0 will be reported.

Related Parameter: **check_tmlib_voltage_map_definition**

CDL

To check the CDL library files, click the GUI menu ***Inspection → Check → CDL***.

Or just right click the session name, and choose ***Check → CDL***.




```
qualib > check -view cdl  -session  sessionName
```

According to the control parameter setting of "CDL Check", enabled checking items will be performed.

cdl_global_nodes

Check Item: check if global nodes have been defined.

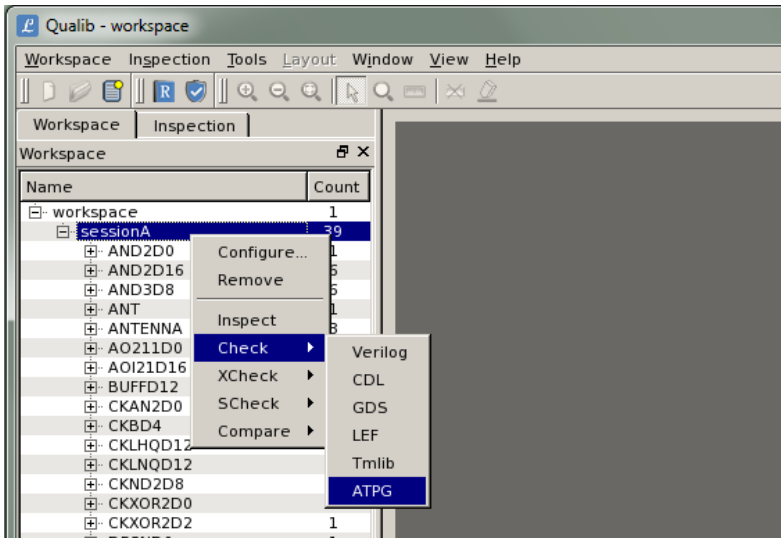
Global nodes will build connections globally between different circuits, and will bring potential risks in designs. As a good style, all the connections should be built through the ports defined in circuits.

Related Parameter: **check_cdl_global_nodes**

ATPG

To check the ATPG library files, click the GUI menu ***Inspection → Check → ATPG***.

Or just right click the session name, and choose ***Check → ATPG***.




```
qualib > check -view atpg  -session  sessionName
```

According to the control parameter setting of "ATPG Check", enabled checking items will be performed.

non_scan_model

Check Item: check if cell non_scan_model is not correct.

Error will be reported for the following situations:

-- The specified non_scan_model does not exist.

-- Number of connections mismatch with number of pins on non_scan_model.

-- The number of bits mismatches for any pin.

-- The pin does not connect to the pin with same name on non_scan_model if it does exist.

Related Parameter: **check_atpg_non_scan_model**

scan_definition_presence

Check Item: check if scan definition is missing or redundant.

For scan cells, which are defined by *scan_cell_name_patterns*, the scan definition must be defined; otherwise it will be reported as missing.

For the other cells, if the scan definition exists, it will be reported as redundant.

Related Parameter: **check_atpg_scan_definition_presence**
scan_cell_name_patterns

5 Cross-Checking

This section describes the cross-checking functionalities between different library view data files.

General

There are some general cross-checking items for each library view.

cell_area

Check Item: cross-check cell area of the specified library views.

Related Parameter: **xcheck_cell_area**
area_error_threshold
gds_boundary_layer

cell_boundary

Check Item: cross-check cell boundary of the specified library views.

Related Parameter: **xcheck_cell_boundary**
gds_boundary_layer

extra_cell

Check Item: cross-check if there are extra cells for the specified library views.

Related Parameter: **xcheck_extra_cells**

missing_cell

Check Item: cross-check if there are missing cells for the specified library views.

Related Parameter: **xcheck_missing_cells**

pin_direction

Check Item: cross-check the pin directions for the specified library views.

Related Parameter: **xcheck_pin_direction**

pin_name

Check Item: cross-check the pin names for the specified library views.

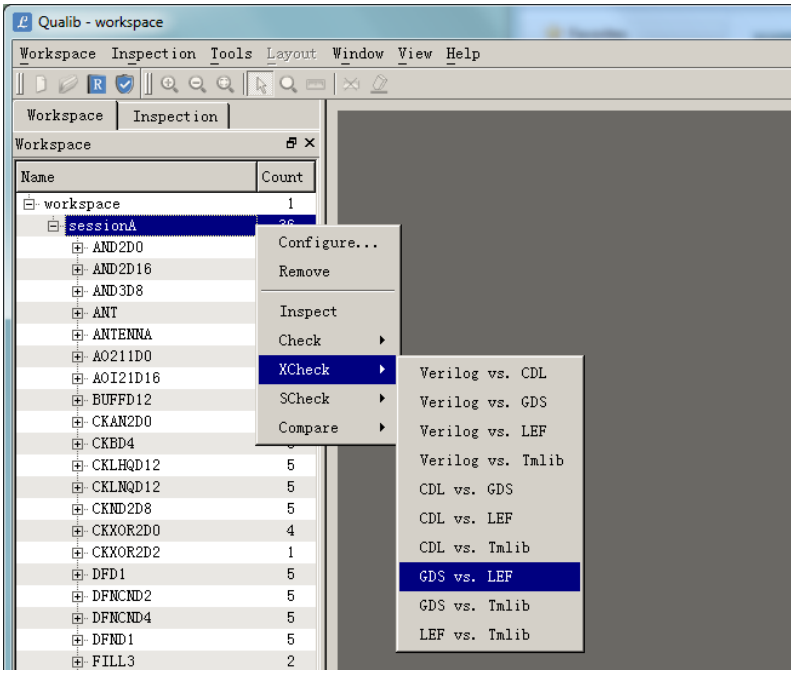
The GDS and CDL internal pins which are specified by the parameter *internal_pin_name_patterns*, will not be reported.

Related Parameter: **xcheck_pin_name**
gds_label_layer_map
internal_pin_name_patterns

GDS vs. LEF

To cross-check between the GDS and LEF library files, click the GUI menu ***Inspection → XCheck → GDS vs. LEF***.

Or just right click the session name, and choose ***XCheck → GDS vs. LEF***.



qualib > **xcheck** *-view_pair* {gds lef} *-session* sessionName

According to the control parameter setting of "GDS vs. LEF Cross-Check", enabled cross-checking items will be performed.

gds_lef_metal_layers

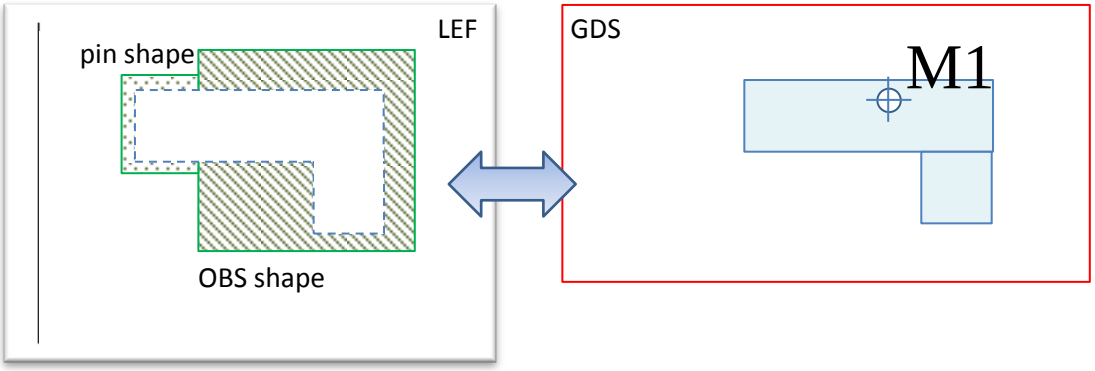
Check Item: cross check if metal layers in LEF design and GDS design are not matched.

If the GDS layer name is different with the LEF layer name, please specify the LEF GDS layer map by the parameter *gds_lef_layer_map*.

Related Parameter: **xcheck_gds_lef_metal_layers**
gds_lef_layer_map

gds_lef_obs_overlap

Check Item: cross check if any GDS metal shape is not covered by LEF OBS shapes and LEF pin shapes.



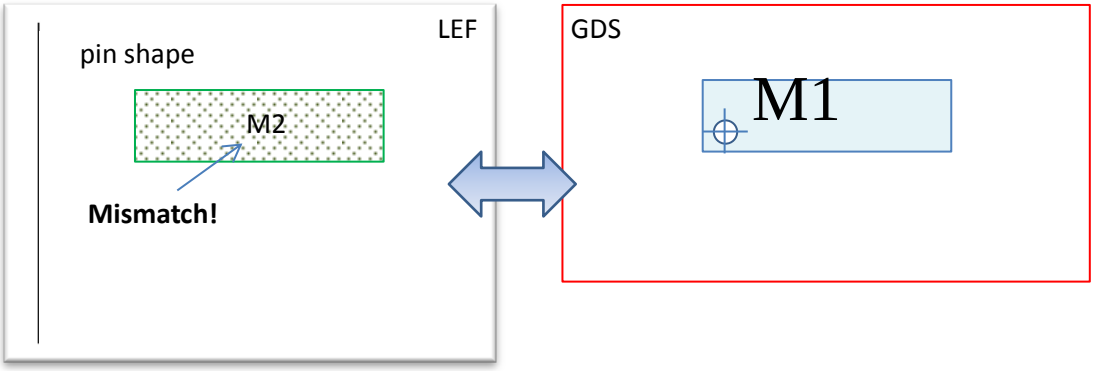
The GDS shapes on the dummy layers specified by the parameter *gds_dummy_layers* will be skipped for this check.
If the parameter *gds_pin_obs_spacing* is specified, the LEF pin shapes will be expanded by the spacing value. The off side threshold is specified by the parameter *geometry_off_side_threshold*. If the shape difference is less than the threshold, it will be skipped to report, e.g. due to alignment on grid, or converting the polygon to rectangles.

If the GDS layer name is different with the LEF layer name, please specify the LEF GDS layer map by the parameter *gds_lef_layer_map*.

Related Parameter: **xcheck_gds_lef_obs_overlap**
gds_dummy_layers
gds_pin_obs_spacing
geometry_off_side_threshold
gds_lef_layer_map

gds_lef_pin_layer

Check Item: cross check if any LEF pin layer is not covered by the GDS pin layers.



If parameter *gds_lef_strict_pin_layer_check* is turned on, the LEF pin layers must be completely matched with the GDS pin layers; otherwise the pin shapes only on LEF layer will be reported.

If parameter *gds_trace_for_pg/signal_pin* is turned on, geometries of P/G pin or signal pin on different layers will be connected by tracing.
The max trace depth is specified by parameter *gds_geometry_search_depth*.
The layer connection rule is specified by the tech file.

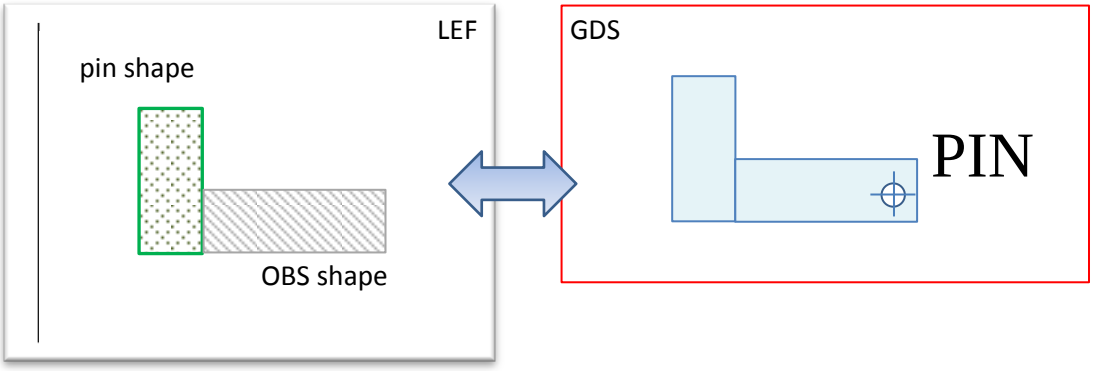
If the label layer is different with the pin layer, please specify the pin label layer map by the parameter *gds_label_layer_map*.

If the GDS layer name is different with the LEF layer name, please specify the LEF GDS layer map by the parameter *gds_lef_layer_map*.

Related Parameter: **xcheck_gds_lef_pin_layer**
gds_trace_for_pg_pin
gds_trace_for_signal_pin
gds_geometry_search_depth
gds_lef_strict_pin_layer_check
gds_label_layer_map
gds_lef_layer_map

gds_lef_pin_shape

Check Item: cross check if any LEF pin shape is not covered by the corresponding GDS pin shapes.



If parameter *gds_lef_strict_pin_shape_check* is turned on, the LEF pin layers must be completely matched with the GDS pin layers; otherwise the pin shapes only on LEF layer will be reported. The off side threshold is specified by the parameter *geometry_off_side_threshold*. If the shape difference is less than the threshold, it will be skipped to report, e.g. due to alignment on grid, or converting the polygon to rectangles.

If parameter *gds_trace_for_pg/signal_pin* is turned on, geometries of P/G pin or signal pin on different layers will be connected by tracing.
The max trace depth is specified by parameter *gds_geometry_search_depth*.
The layer connection rule is specified by the tech file.

If the label layer is different with the pin layer, please specify the pin label layer map by the parameter *gds_label_layer_map*.

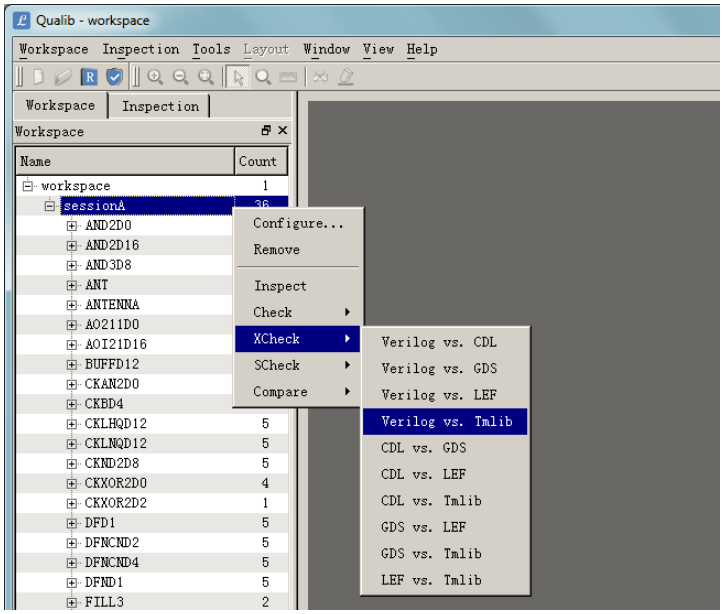
If the GDS layer name is different with the LEF layer name, please specify the LEF GDS layer map by the parameter *gds_lef_layer_map*.

Related Parameter: **xcheck_gds_lef_pin_shape**
gds_lef_strict_pin_shape_check
geometry_off_side_threshold
gds_trace_for_pg_pin
gds_trace_for_signal_pin
gds_geometry_search_depth
gds_label_layer_map
gds_lef_layer_map

Verilog vs. Timing Lib

To cross-check between the verilog netlist files and the timing liberty files, click the GUI menu ***Inspection → XCheck → Verilog vs. Tmlib***.

Or just right click the session name, and choose ***XCheck → Verilog vs. Tmlib***.

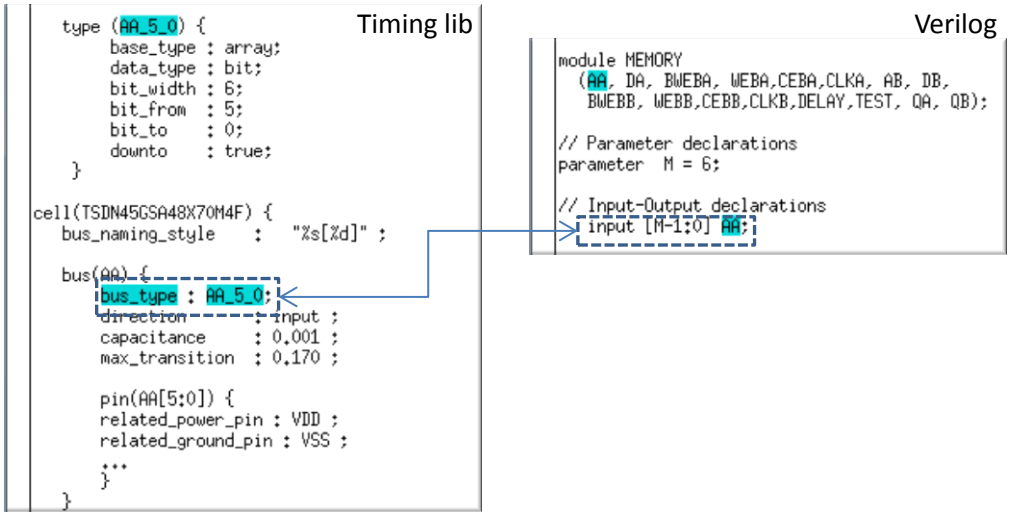


qualib > **xcheck** *-view_pair* {verilog tmlib} *-session* sessionName

According to the control parameter setting of "Verilog vs. Timing Lib Cross-Check", enabled cross-checking items will be performed.

verilog_tmlib_bus

Check Item: cross check if any bus pin in timing library is not matched with Verilog.

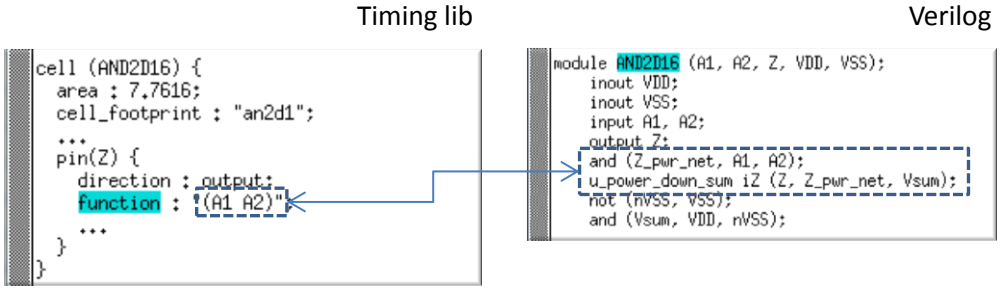


If there is any bus which definition is different between in timing library and in Verilog, the mismatched bus will be reported.

Related Parameter: **xcheck_verilog_tmlib_bus**

verilog_tmlib_pin_function

Check Item: cross check if any pin function in timing library is not matched with Verilog.

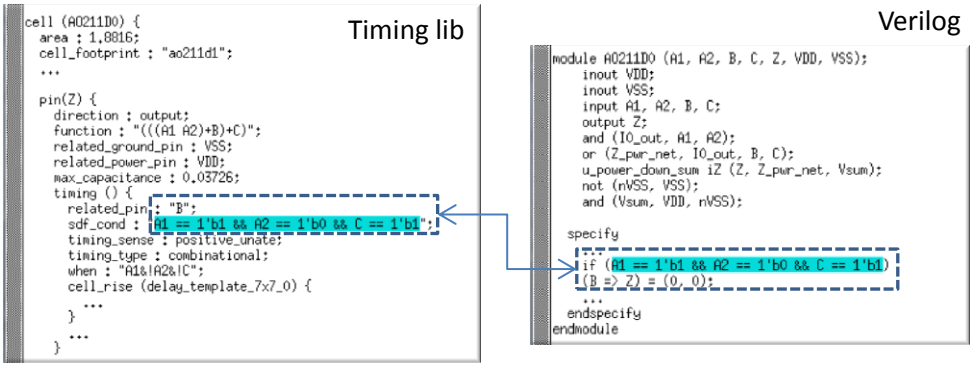


If there is any pin which the function defined in timing library is not matched with defined in Verilog, the mismatched pin functions will be reported.

Related Parameter: **xcheck_verilog_tmlib_pin_function**

verilog_tmlib_timing_arc

Check Item: cross check if any timing arc in timing library is not matched with Verilog.

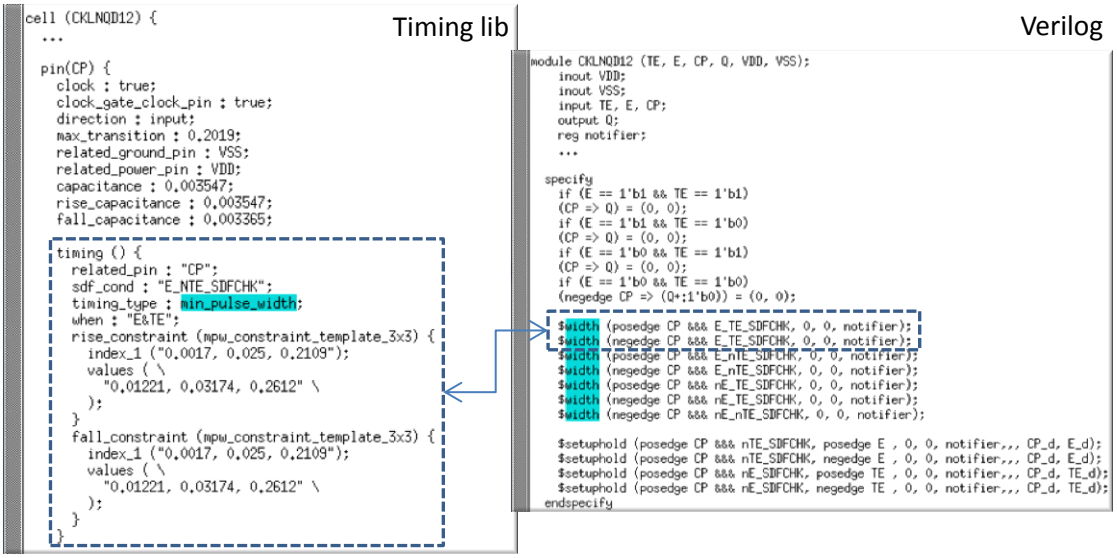


If there is any mismatch, the mismatched timing arc in timing library and Verilog will be reported.
If no "sdf_cond" defined, "*" will be reported as default "sdf_cond".

Related Parameter: **xcheck_verilog_tmlib_timing_arc**

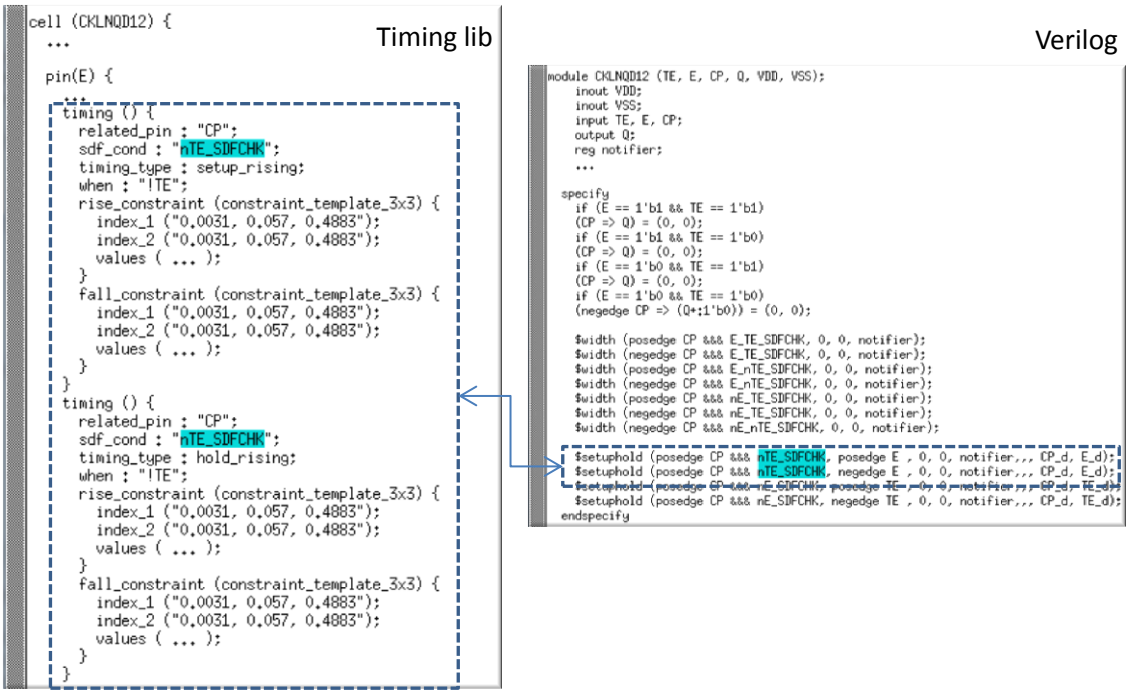
verilog_tmlib_timing_constraint

Check Item: cross check if any timing check in timing library is not matched with Verilog.



Following timing types of timing check definitions will be compared.

in Timing lib	in Verilog
min_pulse_width	$width
setup{hold}_rising{falling}	$setuphold, $setup, $hold
recovery{removal}_rising{falling}	$recrem, $removal, $recovery



If no “sdf_cond” defined, “*” will be reported as default “sdf_cond”.

Related Parameter: **xcheck_verilog_tmlib_timing_constraint**

6 Self-Checking

This section describes the cross-checking functionalities of same library view files in different corners, which is called “Self-Checking”.

General

There are some general checking items of self-checking for each library view.

cell_area

Check Item: check if the cell area is not matched with the reference corner for a specified view.

Related Parameter: **scheck_cell_area**
area_error_threshold

cell_boundary

Check Item: check if cell boundary is not matched with the reference corner for a specified view.

Related Parameter: **scheck_cell_boundary**
gds_boundary_layer

extra_cells

Check Item: check if there is any extra cell than the reference corner for a specified view.

Related Parameter: **scheck_extra_cells**

missing_cells

Check Item: check if there is any missing cell than the reference corner for a specified view.

Related Parameter: **scheck_missing_cells**

pin_direction

Check Item: check if pin directions are not matched with the reference corner for a specified view.

Related Parameter: **scheck_pin_direction**

pin_name

Check Item: check if pin names are not matched with the reference corner for a specified view.

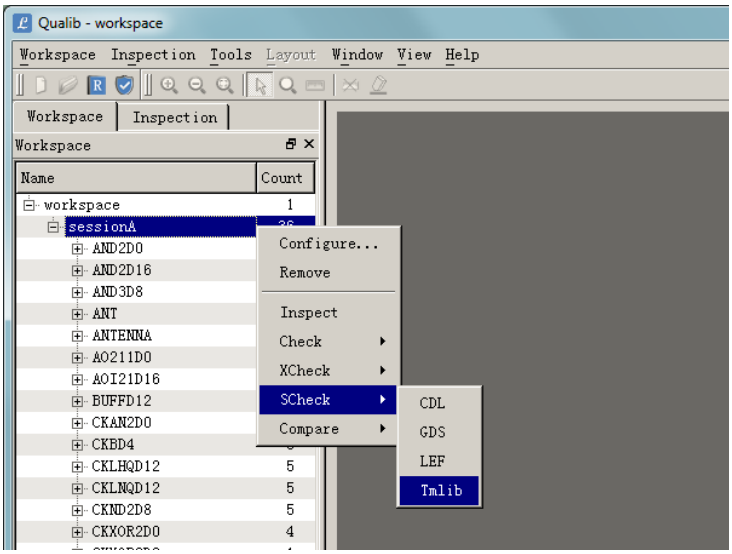
The GDS and CDL internal pins which are specified by the parameter *internal_pin_name_patterns*, will not be reported.

Related Parameter: **scheck_pin_name**
gds_label_layer_map
internal_pin_name_patterns

Timing Lib

To self-check between the Timing library files in different corners, click the GUI menu ***Inspection → SCheck → Tmlib***.

Or just right click the session name, and choose ***SCheck → Tmlib***.




```
qualib > scheck  -view tmlib  -session  sessionName
```

According to the control parameter setting of "Timing Lib Self-Check", enabled cross-checking items will be performed.

tmlib_leakage_power_group

Check Item: check if any leakage power group is not matched with the reference corner.

```
cell (AO211D0) {
  area : 1.8816;
  cell_footprint : "ao211d1";
  ...
  leakage_power () {
    value : 161.216550;
    when : "!A1 !A2 !B !C !Z";
    related_pg_pin : VDD;
  }
  leakage_power () {
    value : 62.598571;
    when : "!A1 !A2 !B C Z";
    related_pg_pin : VDD;
  }
  ...
}
```

If there is any mismatch, the extra and missing leakage power conditions in different corners will be reported.

Related Parameter: **scheck_tmlib_leakage_power_group**

tmlib_pin_function

Check Item: check if any pin function is not matched with the reference corner.

```
cell (AND3D8) {
  area : 5.6448;
  cell_footprint : "an3d1";

  pin(Z) {
    direction : output;
    function : "((A1 A2) A3)";
    related_ground_pin : VSS;
    related_power_pin : VDD;
    max_capacitance : 0.5962;
    timing () {
      ...
    }
  }
```

If there is any mismatch, the reference and target pin functions in different corners will be reported.

Related Parameter: **xcheck_tmlib_tmlib_pin_function**

tmlib_power_arc

Check Item: check if any power arc is not matched with the reference corner.

A power arc is usually defined in tables: *rise_power* and *fall_power*, from input pin to output pin. If any type of table is not matched with the reference corner, the mismatched power arc will be reported.

If there is any mismatch, the extra and missing power arc in different corners will be reported.

Related Parameter: **xcheck_tmlib_tmlib_power_arc**

tmlib_timing_arc

Check Item: check if any timing arc is not matched with the reference corner.

A timing arc is usually defined in tables: *cell_fall*, *cell_rise*, *fall_transition* and *rise_transition*, from input pin to output pin. If any type of table is not matched with the reference corner, the mismatched timing arc will be reported.

If there is any mismatch, the extra and missing timing arc in different corners will be reported.

Related Parameter: **xcheck_tmlib_tmlib_timing_arc**

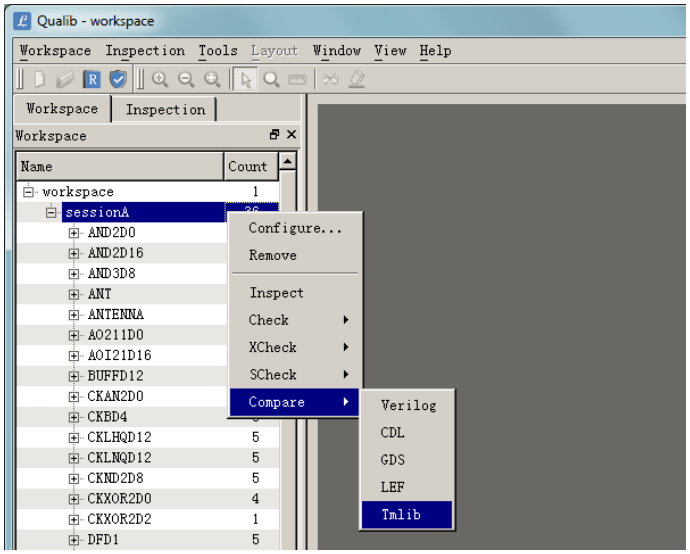
7 Comparison

This section describes how to compare two sets of library files.

Timing Lib

To compare two set of timing library files, click the GUI menu ***Inspection → Compare → Timing Lib***.

Or just right click the session name, and choose ***Compare → Timing Lib***.

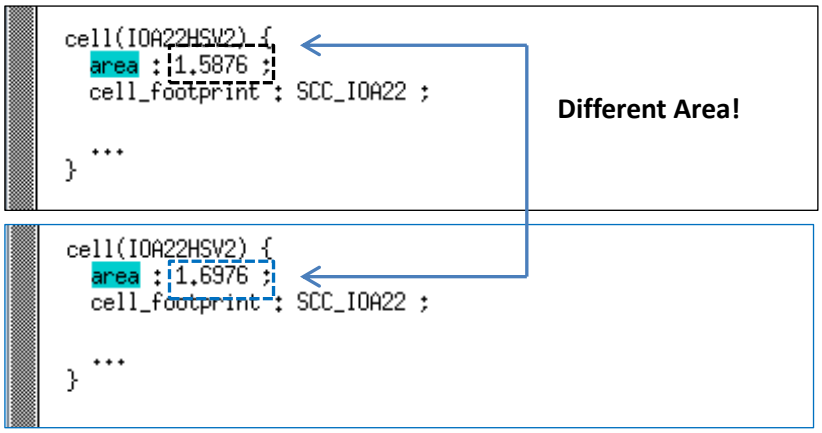



```
qualib > compare -view tmlib   -source_session   s1   -source_corner   c1 \
                               -target_session   s1   -target_corner   c2   *
```

According to the control parameter setting of "Timing Lib Comparing", enabled comparing items will be performed.

area

Check Item: compare area of two library cells.



Related Parameter: **compare_tmlib_area**

hold_constraint

Check Item: compare hold constraint of two library cells.

```
pin(D) {
  direction : input;
  capacitance : 0.0008948;
  ...
  timing () {
    related_pin : "CP";
    timing_type : hold_rising;
    rise_constraint (constraint_template_3x3) {
      index_1 ("0.0017, 0.025, 0.2119");
      index_2 ("0.0017, 0.025, 0.2109");
      values ( \
          "0.002154, -0.001256, -0.003885", \
          "0.004906, 0.002388, -0.001113", \
          "0.005126, 0.003256, -0.005589" \
      );
    }
    fall_constraint (constraint_template_3x3) {
      index_1 ("0.0017, 0.025, 0.2109");
      index_2 ("0.0017, 0.025, 0.2119");
      values ( \
          "0.009001, 0.005563, 0.006147", \
          "0.01272, 0.01143, 0.01007", \
          "0.01867, 0.01671, 0.02138" \
      );
    }
  }
```

A hold constraint group is usually defined by *hold_rising* and *hold_falling*, from clock pin to input pin. In these timing groups, two constraint tables are defined: *rise_constraint* and *fall_constraint*. The hold constraint will be calculated using interpolation with the reference indexes.

Related Parameter: **compare_tmlib_hold_constraint**

internal_power

Check Item: compare internal power of two library cells.

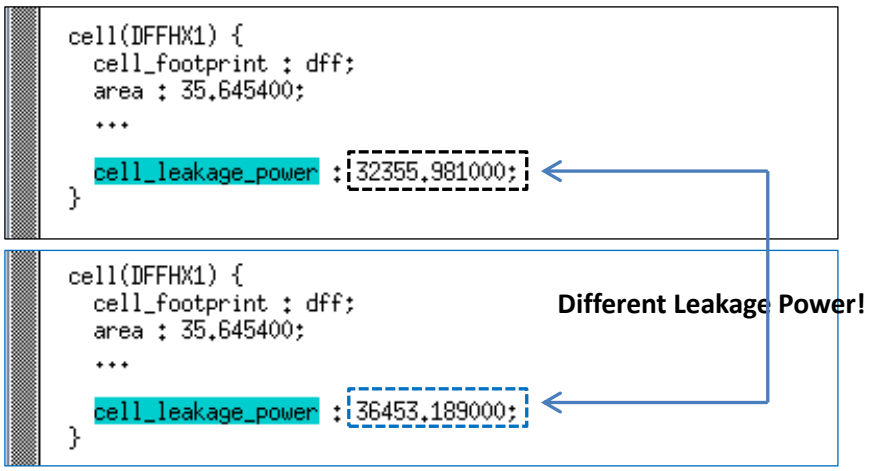
```
pin(A1) {
  direction : input;
  capacitance : 0.002311;
  ...
  internal_power () {
    when : "A2&!A3&!Z";
    related_pg_pin : VDD;
    rise_power (passive_power_template_7x1_0) {
      index_1 ("0.0017, 0.005, 0.0117, 0.025, 0.0515, 0.1047, 0.2109");
      values ( \
        "0.0001144, 0.0001175, 0.0001312, 0.0001154, 0.0001021, 8.378e-05, 3.959e-05" \
      );
    }
    fall_power (passive_power_template_7x1_0) {
      index_1 ("0.0017, 0.005, 0.0117, 0.025, 0.0515, 0.1047, 0.2109");
      values ( \
        "0.001724, 0.001721, 0.001709, 0.001718, 0.001723, 0.001724, 0.001714" \
      );
    }
  }
```

A “internal_power()” group is usually defined in tables: *rise_power* and *fall_power*, from input pin to output pin. Those power values will be calculated by interpolation with the reference indexes.

Related Parameter: **compare_tmlib_internal_power**

leakage_power

Check Item: compare leakage power of two library cells.

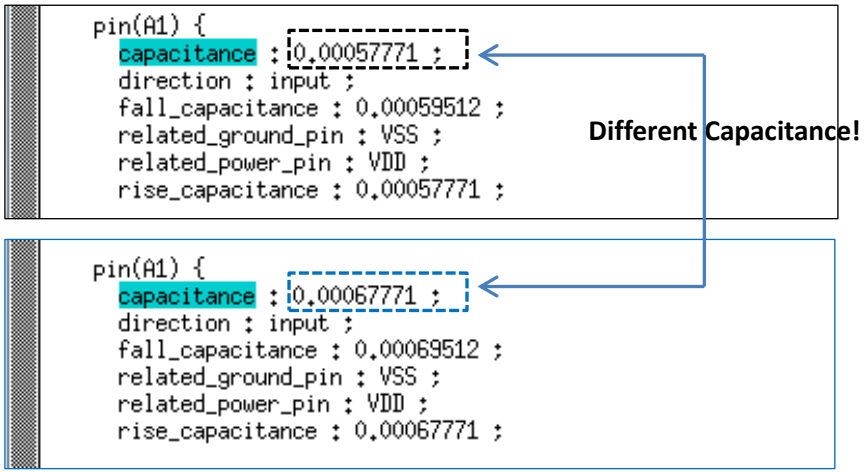


The cell leakage power value is defined by cell_leakage_power field or leakage_power() group.

Related Parameter: **compare_tmlib_leakage_power**

pin_capacitance

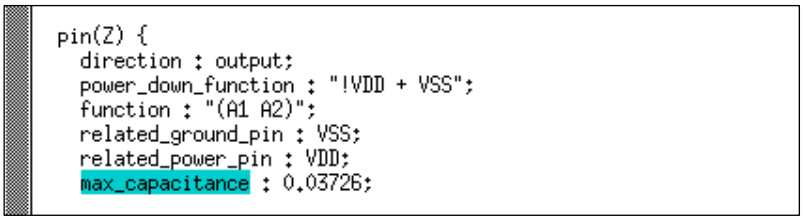
Check Item: compare input pin capacitance of two library cells.



Related Parameter: **compare_tmlib_pin_capacitance**

pin_max_capacitance

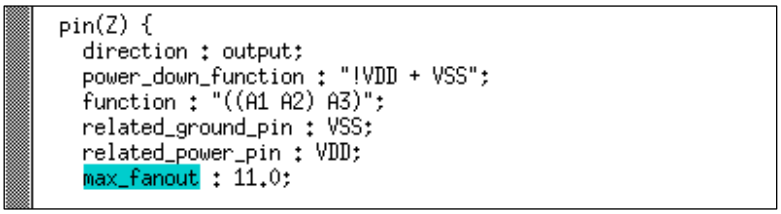
Check Item: compare output pin max_capacitance of two library cells.



Related Parameter: **compare_tmlib_pin_max_capacitance**

pin_max_fanout

Check Item: compare output pin max_fanout of two library cells.



Related Parameter: **compare_tmlib_pin_max_fanout**

pin_max_transition

Check Item: compare input pin max_transition of two library cells.

```
pin(A1) {
  direction : input;
  max_transition : 0.2019;
  related_ground_pin : VSS;
  related_power_pin : VDD;
  capacitance : 0.0005595;
  rise_capacitance : 0.0005324;
  fall_capacitance : 0.0005595;
```

Related Parameter: **compare_tmlib_pin_max_transition**

pin_min_capacitance

Check Item: compare output pin min_capacitance of two library cells.

```
pin("paddr[8]") {
  direction : output ;
  max_transition : 0.210900 ;
  min_transition : 0.000000 ;
  max_capacitance : 0.136200 ;
  min_capacitance : 0.000000 ;
  capacitance : 0.004953 ;
```

Related Parameter: **compare_tmlib_pin_min_capacitance**

recovery_constraint

Check Item: compare recovery constraint of two library cells.

```
pin(CDN) {
  direction : input;
  capacitance : 0.001591;
  ...
  timing () {
    related_pin : "CP";
    sdf_cond : "D_SE_SI_SDFCHK";
    timing_type : recovery_rising;
    when : "D&SE&SI";
    rise_constraint (constraint_template_3x3) {
      index_1 ("0.0017, 0.025, 0.2109");
      index_2 ("0.0017, 0.025, 0.2109");
      values ( \
          "-0.01724, -0.01306, 0.0312", \
          "-0.02301, -0.01708, 0.02072", \
          "-0.02964, -0.02606, 0.006054" \
      );
    }
  }
```

A recovery constraint group is usually defined by *recovery_rising* and *recovery_falling*, from clock pin to input pin. In these timing groups, two constraint tables are defined: *rise_constraint* and *fall_constraint*. The recovery constraint will be calculated using interpolation with the reference indexes.

Related Parameter: **compare_tmlib_recovery_constraint**

removal_constraint

Check Item: compare removal constraint of two library cells.

```
pin(CDN) {
  direction : input;
  capacitance : 0.001591;
  ...
  timing () {
    related_pin : "CP";
    sdf_cond : "D_SE_SI_SDFCHK";
    timing_type : removal_rising;
    when : "D&SE&SI";
    rise_constraint (constraint_template_3x3) {
      index_1 ("0.0017, 0.025, 0.2109");
      index_2 ("0.0017, 0.025, 0.2109");
      values ( \
          "0.03421, 0.03697, 0.06823", \
          "0.03967, 0.0429, 0.07423", \
          "0.05157, 0.05437, 0.08374" \
      );
    }
  }
```

A removal constraint group is usually defined by *removal_rising* and *removal_falling*, from clock pin to input pin. In these timing groups, two constraint tables are defined: *rise_constraint* and *fall_constraint*. The removal constraint will be calculated using interpolation with the reference indexes.

Related Parameter: **compare_tmlib_removal_constraint**

setup_constraint

Check Item: compare setup constraint of two library cells.

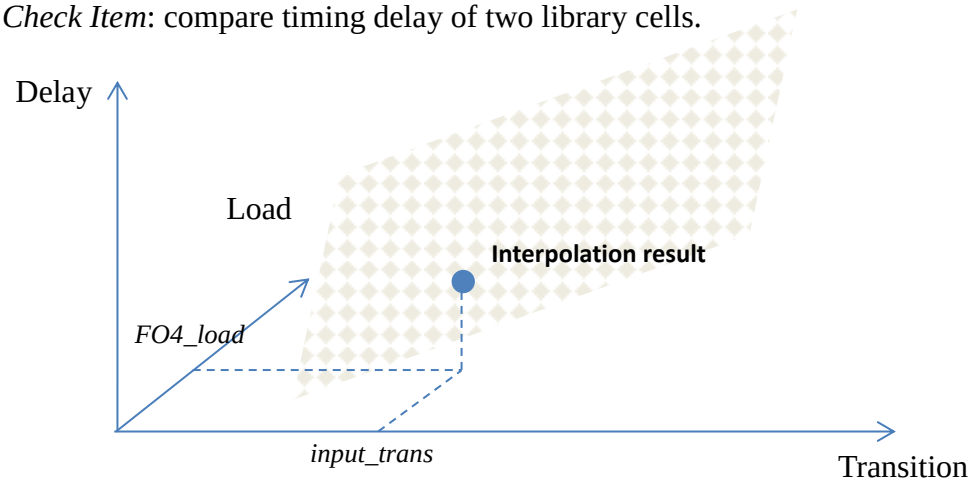
```
pin(TE) {
  direction : input;
  capacitance : 0.0008892;
  ...
  timing () {
    related_pin : "CP";
    sdf_cond : "nE_SDFCHK";
    timing_type : setup_rising;
    when : "!E";
    rise_constraint (constraint_template_3x3) {
      index_1 ("0.0017, 0.025, 0.2109");
      index_2 ("0.0017, 0.025, 0.2109");
      values ( \
          "0.02399, 0.02776, 0.03685", \
          "0.02157, 0.02525, 0.03347", \
          "0.03718, 0.04008, 0.04752" \
      );
    }
    fall_constraint (constraint_template_3x3) {
      index_1 ("0.0017, 0.025, 0.2109");
      index_2 ("0.0017, 0.025, 0.2109");
      values ( \
          "0.01847, 0.02266, 0.04424", \
          "0.01431, 0.0185, 0.03958", \
          "0.01018, 0.0149, 0.03402" \
      );
    }
  }
```

A setup constraint group is usually defined by *setup_rising* and *setup_falling*, from clock pin to input pin. In these timing groups, two constraint tables are defined: *rise_constraint* and *fall_constraint*. The setup constraint will be calculated using interpolation with the reference indexes.

Related Parameter: **compare_tmlib_setup_constraint**

timing_delay

Check Item: compare timing delay of two library cells.

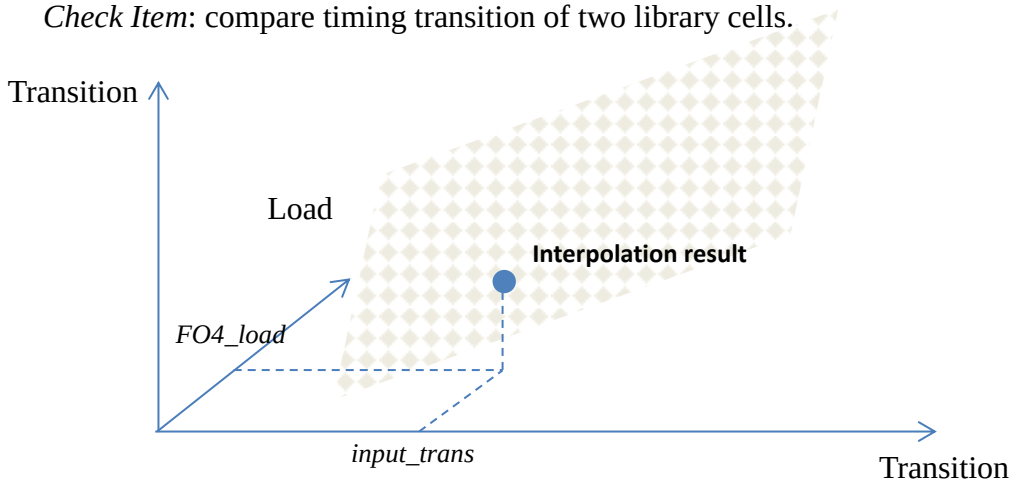


A "timing()" group is usually defined in tables: *cell_fall*, *cell_rise*, *fall_transition* and *rise_transition*, from input pin to output pin. Those arc values will be calculated by FO4 interpolation for output delay or transition (using the output transition of ideal input as input transition; using 4X input capacitance as output load).

Related Parameter: **compare_tmlib_timing_delay**

timing_transition

Check Item: compare timing transition of two library cells.



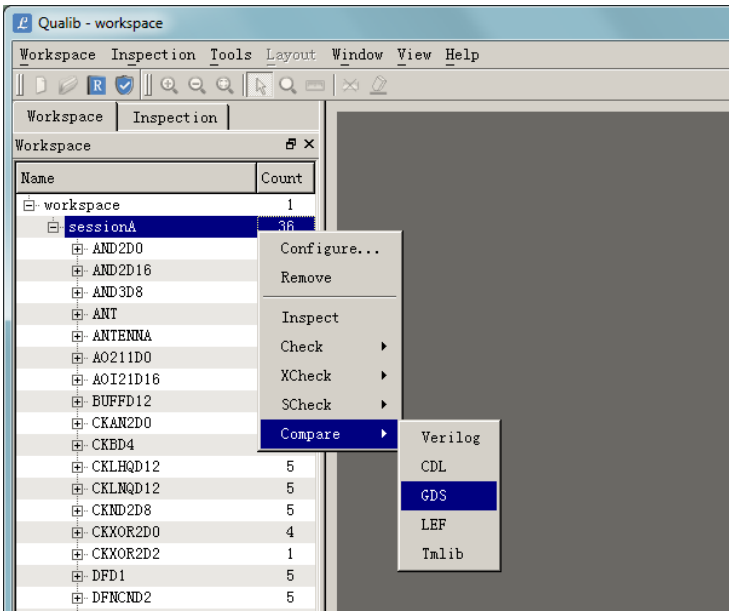
A "timing()" group is usually defined in tables: *cell_fall*, *cell_rise*, *fall_transition* and *rise_transition*, from input pin to output pin. Those arc values will be calculated by FO4 interpolation for output delay or transition (using the output transition of ideal input as input transition; using 4X input capacitance as output load).

Related Parameter: **compare_tmlib_timing_transition**

GDS

To compare two set of GDS library files, click the GUI menu ***Inspection*** → ***Compare*** → ***GDS***.

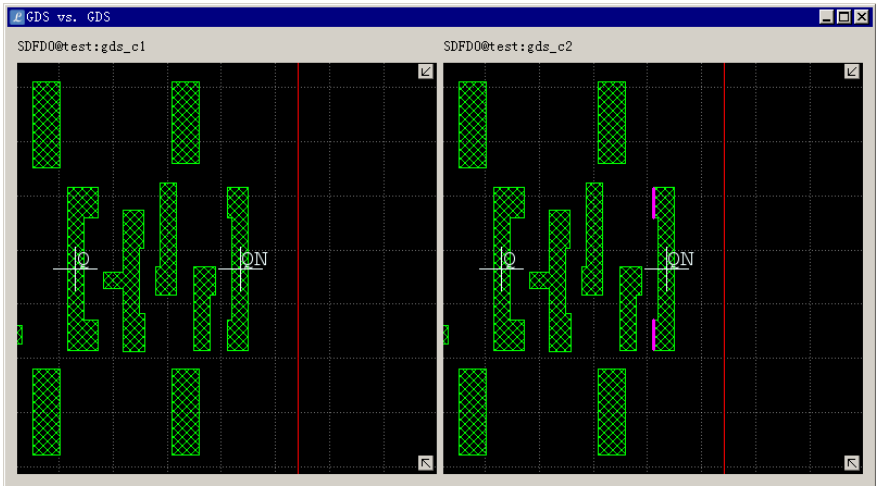
Or just right click the session name, and choose ***Compare*** → ***GDS***.




```
qualib > compare -view gds   -source_session   s1   -source_corner   c1 \
                             -target_session   s1   -target_corner   c2   *
```

cell_layout

Check Item: compare the GDS layouts.

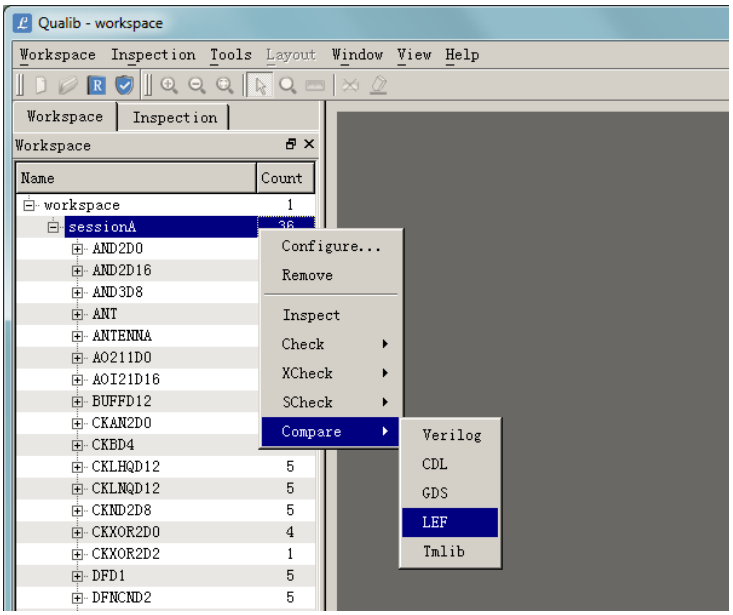


Related Parameter: **compare_gds_cell_layout**

LEF

To compare two set of LEF library files, click the GUI menu ***Inspection*** → ***Compare*** → ***LEF***.

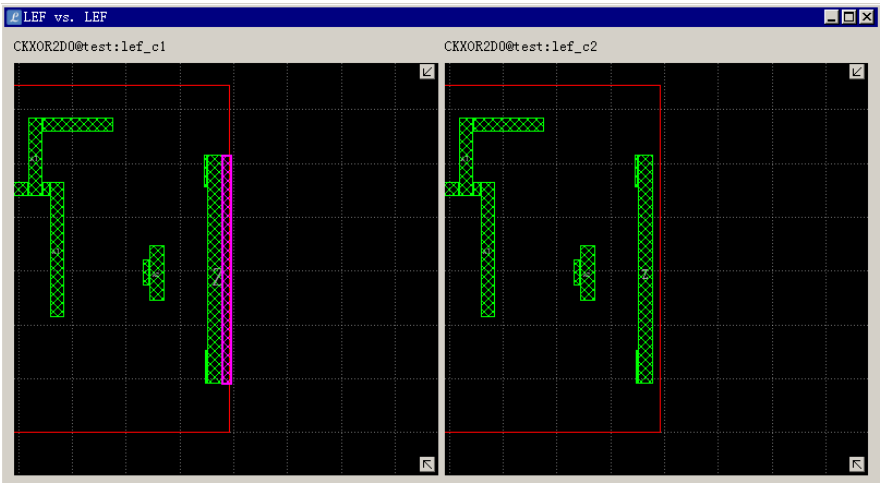
Or just right click the session name, and choose ***Compare*** → ***LEF***.




```
qualib > compare -view lef    -source_session   s1   -source_corner   c1 \
                              -target_session   s1   -target_corner   c2    *
```

cell_layout

Check Item: compare the LEF layouts.



Related Parameter: **compare_lef_cell_layout**

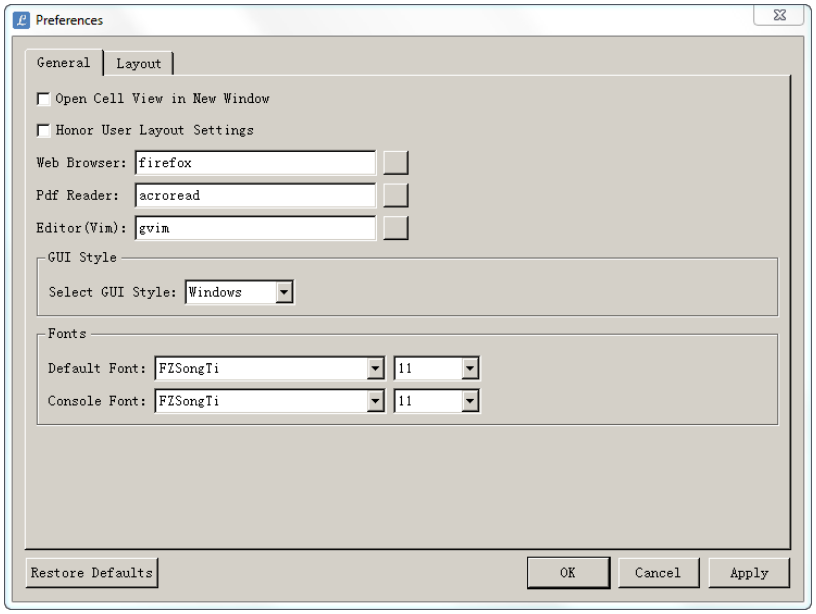
8
Setting Preferences

This section describes how to set the preferences.

General Preferences

To change the general preferences, click the GUI menu ***Tools → Preferences…***

In the "***General***" tab page, there are following preferences defined.



Click ***Apply*** button to commit the change.

Click ***Restore Defaults*** button to restore the original preference setting.

- *Open Cell View in New Window*

  If it is checked, tools will open multiple LEF or GDS viewer windows.

- *Honor User Layout Settings*

  If it is checked, LEF or GDS viewer windows will honor those user layer settings. On default, when user double clicks the check result, the related LSW settings will be turned on automatically.

- *Web Browser*

It is used to define the web browser to open the “Visit Us” web site.

- *Pdf Reader*

  It is used to define the PDF reader to open the User Guide document.

- *Editor*

  It is used to define the editor to open the text-format library files.

- *GUI Style*

  It is used to define the appearance of GUI windows, icons, and dialogs.

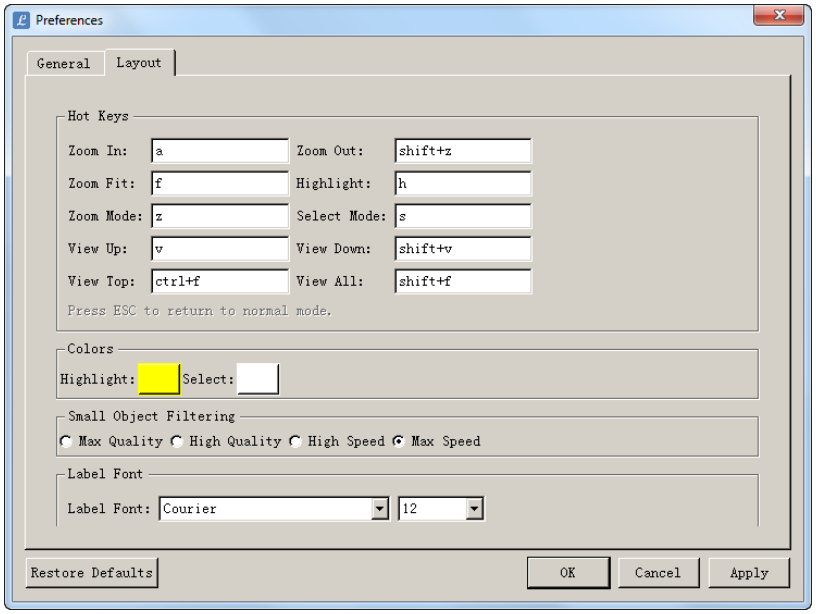
- *Fonts*

  It is used to define the fonts displayed in the window menus and dialogs; and also the fonts displayed in the console.

Layout Preferences

To change the layout preferences, click the GUI menu ***Tools → Preferences…***

In the "***Layout***" tab page, there are following preferences defined.



Click ***Apply*** button to commit the change.

Click ***Restore Defaults*** button to restore the original preference setting.

- *Hot Keys*

  User can define hot keys used in LEF and GDS viewer windows.
  For example, Zoom In, Zoom Out, GDS level of view changing …

- *Colors*

  User can define the highlight color and selections color used in LEF and GDS viewer windows.

- *Small Object Filtering*

  It is used to define the criteria to display those small objects (e.g. pins, vias, cells …) in LEF and GDS viewer windows.

Max Quality	Draw objects with max dimension larger than 1.0 pixel.
High Quality	Draw objects with max dimension larger than 1.5 pixels.
High Speed	Draw objects with max dimension larger than 2.0 pixels.
Max Speed	Draw objects with max dimension larger than 2.5 pixels.

- *Label Font*

  It is used to define the label fonts displayed in GDS viewer window.

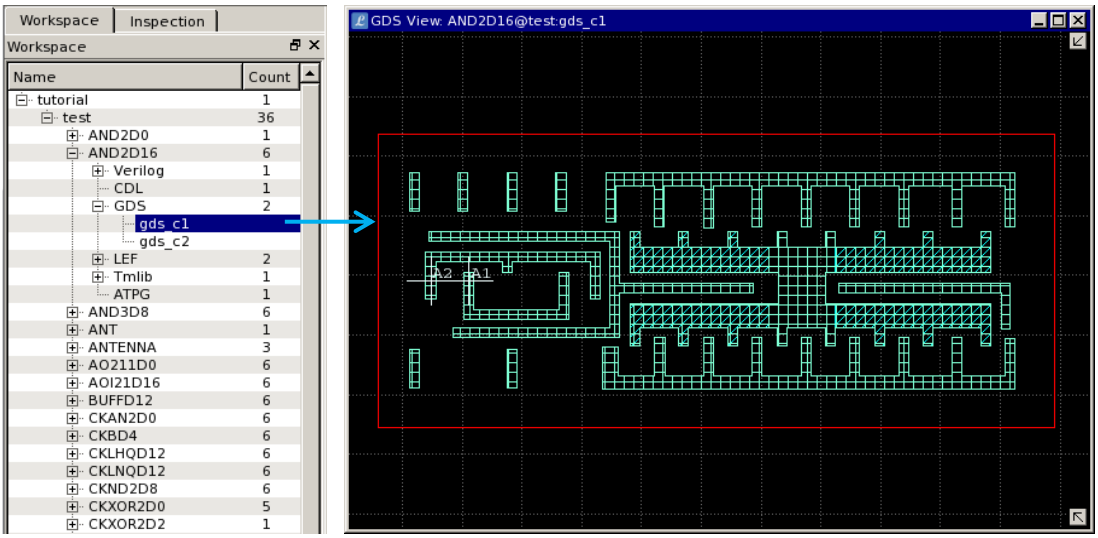
9

Library View Windows

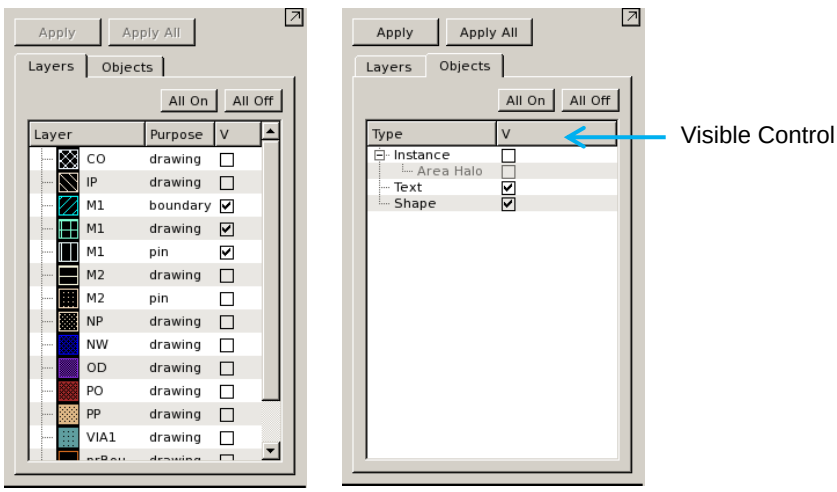
This section describes the library view GUI windows.

GDS View

Double click the GDS view of cell list in the workspace, or double click the GDS related detail check result item, the GDS view GUI window will pop up.

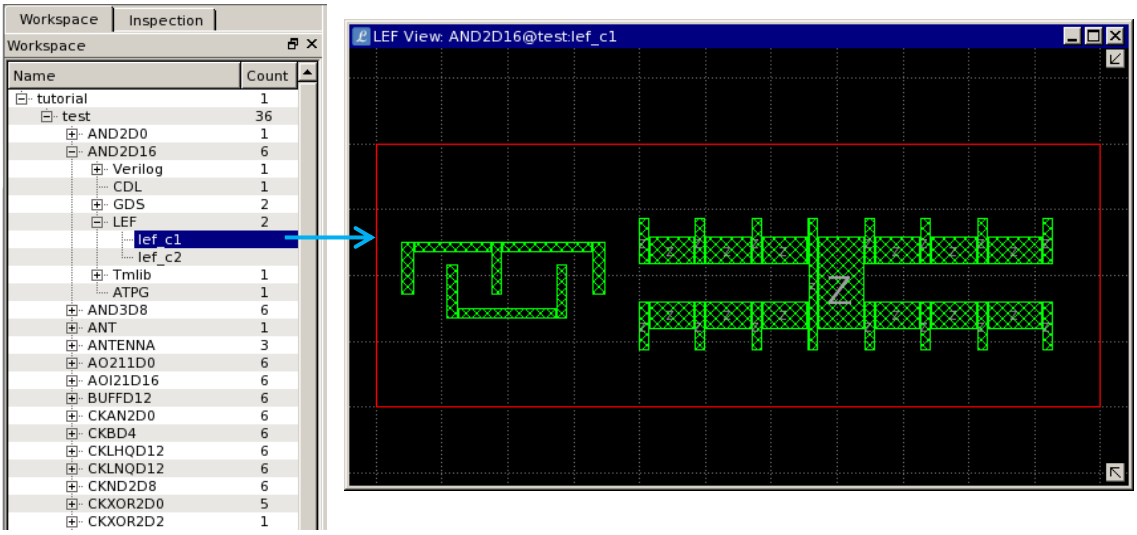


Click the  button in the top right corner, the configuration window will be shown. User can select specific layers or objects to displayed in the canvas.

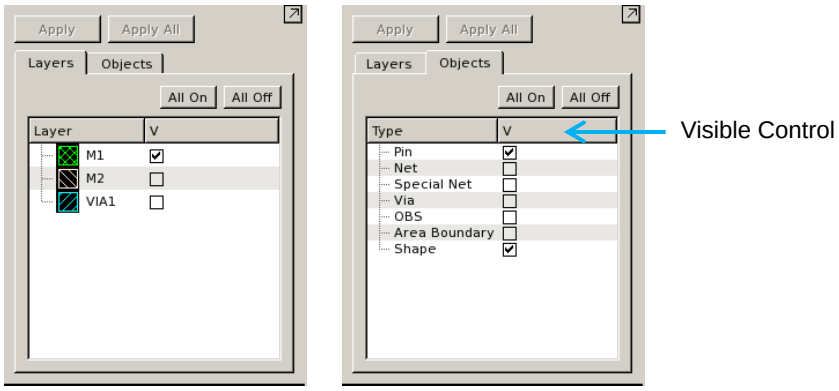


LEF View

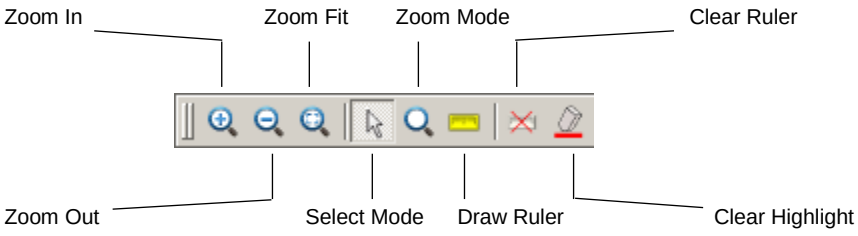
Double click the LEF view of cell list in the workspace, or double click the LEF related detail check result item, the LEF view GUI window will pop up.



Click the  button in the top right corner, the configuration window will be shown. User can select specific layers or objects to displayed in the canvas.

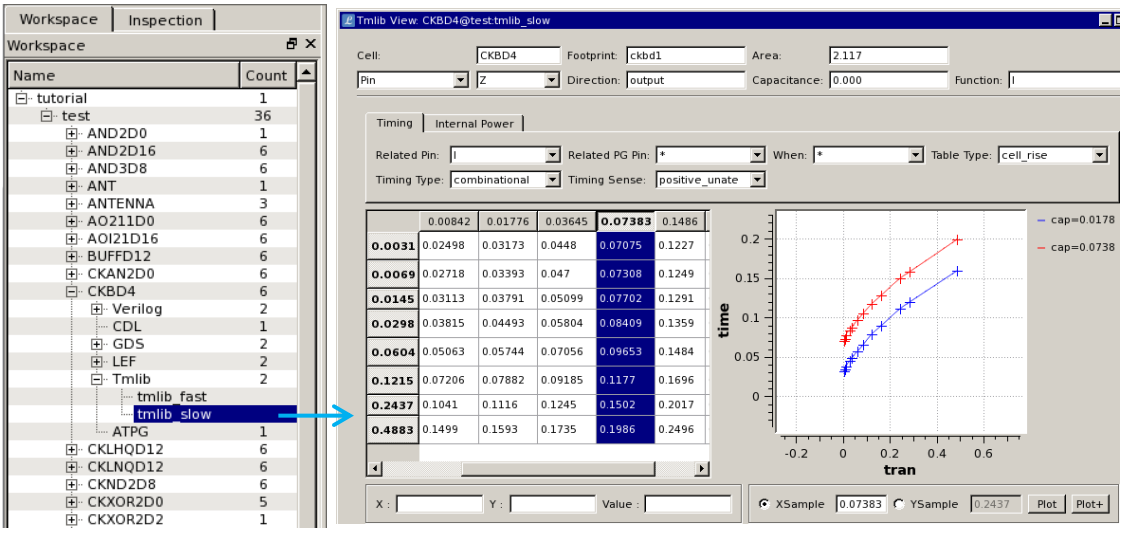


For both of LEF and GDS layout view windows, the toolbar can be used to control the zoom, highlight, and ruler.



Timing Lib View

Double click the Tmlib view of cell list in the workspace, or double click the timing lib related detail check result item, the timing lib view GUI window will pop up.



In timing lib view GUI window, the timing and internal power table contents will be shown. User can select some row or column of the matrix, and click ***Plot*** or ***Plot+*** button to display the curve fitting.