PrimeTime® Constraint Consistency Variables and Attributes

Version T-2022.03-SP4, September 2022

SYNOPSYS®

Copyright Notice and Proprietary Information

Destination Control Statement

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

Disclaimer

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

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at
http://www.synopsys.com/company/legal/trademarks-brands.html.
All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

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

Third-Party Links

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

www.synopsys.com

Contents

b2t_clock_mapping_match_names

Specifies whether to perform name comparison while mapping clocks in Block-to-Top analysis.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

This variable specifies whether or not to compare names of clocks while mapping clocks in the Block-to-Top analysis. When set to *false*, names are not considered to identify clock mapping between Block and Top. When set to *true*, B2T uses name comparison in addition to waveform and source comparison to identify clock mapping between Block and Top.

To determine the current value of this variable, type

```
ptc_shell> printvar b2t_clock_mapping_match_names
```

or

```
ptc_shell> echo $b2t_clock_mapping_match_names
```

SEE ALSO

printvar(2)

b2t_enable_unconstrained_path_comparison

Specifies whether to do behavior comparison between untested and unconstrained paths in Block-to-Top analysis.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

This variable specifies whether or not to compare relations between untested and unconstrained paths in the Block-to-Top analysis. When set to *false*, no violations are flagged between untested and unconstrained paths in B2T analysis. When set to *true*, any applicable external delay violations between untested and unconstrained paths in block-to-top analysis are flagged.

To determine the current value of this variable, enter

```
ptc_shell> printvar b2t_enable_unconstrained_path_comparison
```

or

```
ptc_shell> echo $b2t_enable_unconstrained_path_comparison
```

SEE ALSO

printvar(2)

b2t_report_unused_vclk

Specifies whether to report unused virtual clocks of each block instance in Block-to-Top analysis.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

This variable specifies whether or not to report unused virtual clocks of each block instance in the Block-to-Top analysis. When set to *false*, No B2T_CLK_0014 violations. When set to *true*, B2T analyzes all virtual clocks of the block instance, and reports B2T_CLK_0014 violations if there are any unused Virtual clocks.

To determine the current value of this variable, type

```
ptc_shell> printvar b2t_report_unused_vclk
```

or

```
ptc_shell> echo $b2t_report_unused_vclk
```

SEE ALSO

printvar(2)

b2t_suppress_violations

Specifies whether violations that are only on min or only on max should be suppressed in Block-to-Top analysis.

TYPE

string

DEFAULT

none

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

This variable specifies whether violations that are only on min or only on max corner should be suppressed in the Block-to-Top analysis. Allowed values are *none* (the default), *hold_only*, or *setup_only*. When set to *none*, no partial violations are suppressed in B2T analysis. When set to *hold_only*, violations that are "only" on the "min" corner are suppressed in B2T analysis. And when set to *setup_only*, violations that are "only" on the "max" corner are suppressed in B2T analysis.

To determine the current value of this variable, type

```
ptc_shell> printvar b2t_suppress_violations
```

or

```
ptc_shell> echo $b2t_suppress_violations
```

SEE ALSO

printvar(2)

bus_naming_style

Sets the naming format for a specific element of a bus.

TYPE

string

DEFAULT

%s[%d]

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

This variable is used by the native Verilog reader to set the naming format for a specific element of a bus. This is the way that the names of the individual bits of the bus appear in the application.

The default is "%s[%d]". For example, for bus A and index 12, the name would be A[12].

To determine the current value of this variable, type

```
ptc_shell> printvar bus_naming_style
or
ptc_shell> echo $bus_naming_style
```

SEE ALSO

printvar(2)
read_verilog(2)

case_analysis_propagate_through_icg

Determines whether case analysis is propagated through integrated clock gating cells.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

When this variable is *false* (the default), constants propagating throughout the design stop propagating when an integrated clock gating cell is encountered. Regardless of whether the integrated clock gating cell is enabled or disabled, no logic values propagate in the fanout of the cell.

When this variable is *true*, constants propagated throughout the design propagate through an integrated clock gating cell provided the cell is enabled. An integrated clock gating cell is enabled when its enable pin (or test enable pin) is set to a high logic value. If the cell is disabled, then the disable logic value for the cell is propagated in its fanout. For example, when the latch_posedge integrated clock gating is disabled, it propagates a logic 0 in its fanout.

Since all latch based integrated clock gating cells are sequential in nature, these cells are only considered for logic propagation if the *case_analysis_sequential_propagation* variable is set to *always*.

To activate logic propagation through all integrated clock gating cells, set the following before using the **update_timing** command:

```
ptc_shell> set case_analysis_sequential_propagation always
ptc_shell> set case_analysis_propagate_through_icg true
```

To determine the current value of this variable, type

```
ptc_shell> printvar case_analysis_propagate_through_icg
or
ptc_shell> echo $case_analysis_propagate_through_icg
```

SEE ALSO

case_analysis_sequential_propagation(3)
set_case_analysis(2)
remove_case_analysis(2)

case_analysis_sequential_propagation

Specifies whether case analysis is propagated across sequential cells.

TYPE

string

DEFAULT

never

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

This variable specifies whether case analysis is propagated across sequential cells. Allowed values are *never* (the default) or *always*. When set to *never*, case analysis is not propagated across the sequential cells. When set to *always*, case analysis is propagated across the sequential cells.

To determine the current value of this variable, type

```
ptc_shell> printvar case_analysis_sequential_propagation
or
ptc_shell> echo $case_analysis_sequential_propagation
```

SEE ALSO

printvar(2)
set_case_analysis(2)

cell_attributes

Description of the predefined attributes for cells.

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Attributes are properties assigned to objects such as pins, cells and nets. Definitions for cell attributes are provided in the subsections that follow.

Attributes are informational, or "read-only." You cannot set the value of attributes designed by the application.

To determine the value of an attribute, use the **get_attribute** command. To see a list of all attributes available for a class of objects use the **list_attributes -application** command.

Cell Attributes

area

A float attribute for the area of a cell.

base_name

A string attribute for the name of a cell. This name does not include the instance path prefix.

full_name

A string attribute for the name of a cell. This name includes a prefix for the instance path to this cell.

hold_uncertainty

A float attribute representing the user-specified clock uncertainty for hold. This attribute exists only if **set_clock_uncertainty** is specified on this object.

is_black_box

A Boolean attribute. The value is "true" if this cell is unresolved or if it refers to a lib_cell with no logic function.

is_combinational

A Boolean attribute that is true if the cell is not sequential.

is_design_mismatch

A Boolean attribute that is annotated by the linker indicating this cell has pins that are inconsistent with its master module.

is_fall_edge_triggered

A Boolean attribute that is true if the cell has timing behavior relative to the falling edge of a clock signal.

is_hierarchical

A Boolean attribute. The value is "true" if this cell is hierarchical, "false" otherwise.

is_integrated_clock_gating_cell

A Boolean attribute that is true if the cell refers to a lib_cell that is defined in the library as an integrated clock gating cell.

is_memory_cell

A Boolean attribute that is true if the cell refers to a lib_cell that is defined in the library as a memory cell.

is_mux

A Boolean attribute that is true if the cell is a mux.

is_negative_level_sensitive

A Boolean attribute that is true if the cell has negative level-sensitive timing behavior, as in a negative D-latch.

is_pad_cell

A Boolean attribute that is true if the cell refers to a lib_cell that is defined in the library as a pad cell.

is_positive_level_sensitive

A Boolean attribute that is true if the cell has positive level-sensitive timing behavior, as in a positive D-latch.

is_rise_edge_triggered

A Boolean attribute that is true if the cell has timing behavior relative to the rising edge of a clock signal.

is_sequential

A Boolean attribute that is true if the cell has sequential logic function.

is_three_state

A Boolean attribute that is true if the cell has one or more three_state outputs.

number_of_pins

An integer attribute representing the number of pins on this cell.

object_class

A string attribute with the value of "cell". Each object class has a different object class value. The possible object classes are: design, cell, pin, port, net, lib, lib_cell, lib_pin, clock, scenario, input_delay, output_delay, timing_exception, timing_arc, lib_timing_arc.

ref_name

A string attribute. The value is the base name of the design or lib_cell to which this cell instance refers.

setup_uncertainty

A float attribute representing the user-specified clock uncertainty for setup. This attribute exists only if **set_clock_uncertainty** is specified on this object.

temperature_max

A float attribute that specifies the maximum temperature value, in degrees Celsius, for the cell.

temperature_min

A float attribute that specifies the minimum temperature value, in degrees Celsius, for the cell.

voltage_max

A float attribute that specifies the voltage value, in volts, for the maximum or single operating condition for the cell.

voltage_min

A float attribute that specifies the voltage value, in volts, for the minimum or single operating condition for the cell.

SEE ALSO

get_attribute(2)
list_attributes(2)
set_clock_uncertainty(2)

clock_attributes

Specifies the predefined attributes for clocks.

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Attributes are properties assigned to objects such as pins, cells, and nets. Definitions for clock attributes are provided in the subsections that follow.

Attributes are informational or "read-only." You cannot set the value of attributes designed by the application.

To determine the value of an attribute, use the **get_attribute** command. To see a list of all attributes available for a class of objects use the **list_attributes -application** command.

Clock Attributes

clock_latency_fall_max

Type: float
Represents the user-specified clock network latency for max fall. The value defaults to 0.0.

clock_latency_fall_min

Type: float
Represents the user-specified clock network latency for min fall. The value defaults to 0.0.

clock_latency_rise_max

Type: float
Represents the user-specified clock network latency for max rise. The value defaults to 0.0.

clock_latency_rise_min

Type: float
Represents the user-specified clock network latency for min rise. The value defaults to 0.0.

clock_network_pins

Type: collection
The value is a collection of pins in the network of this clock.

clock_source_latency_early_fall_max

Type: float
Represents the user-specified clock network latency for early max fall.

clock_source_latency_early_fall_min

Type: float
Represents the user-specified clock network latency for early min fall.

clock_source_latency_early_rise_max

Type: float
Represents the user-specified clock network latency for early max rise.

clock_source_latency_early_rise_min

Type: float
Represents the user-specified clock network latency for early min rise.

clock_source_latency_late_fall_max

Type: float
Represents the user-specified clock network latency for late max fall.

clock_source_latency_late_fall_min

Type: float
Represents the user-specified clock network latency for late min fall.

clock_source_latency_late_rise_max

Type: float
Represents the user-specified clock network latency for late max rise.

clock_source_latency_late_rise_min

Type: float
Represents the user-specified clock network latency for late min rise.

clock_source_latency_pins

Type: collection
The value is a collection of pins in the source latency network of this clock.

clock_transition_fall_max

Type: float
Represents the user-specified clock transition for late max fall.

clock_transition_fall_min

Type: float
Represents the user-specified clock transition for early min fall.

clock_transition_rise_max

Type: float
Represents the user-specified clock transition for late max rise.

clock_transition_rise_min

Type: float
Represents the user-specified clock transition for early min rise.

command_text

Type: string
The value is the equivalent commands that can re-create the object.

file_line_info

Type: string
If it exists, the value contains one or more Tcl or SDC source file name and line number pairs of the commands which were used to create this object.

full_name

Type: string
Name of a clock.

generated_clocks

Type: collection
If it exists, the value is a collection containing the generated clocks that this clock is the master clock of.

hold_uncertainty

Type: float
Represents the user-specified clock uncertainty for hold.

is_active

Type: Boolean
The value is "true" if this clock is active, "false" if the clock was excluded by **set_active_clocks**.

is_generated

Type: Boolean
The value is "true" if this clock is a generated clock, "false" otherwise.

master_clock

Type: collection
If it exists, the value is a collection containing the master clock for this generated clock.

master_pin

Type: collection
If it exists, the value is a collection containing the pin given to the -source option for this generated clock.

max_capacitance_clock_path_fall

max_capacitance_clock_path_rise

max_capacitance_data_path_fall

max_capacitance_data_path_rise

A floating point attribute set with **set_max_capacitance**.

max_time_borrow

Type: float
A floating point attribute set with **set_max_time_borrow**.

max_transition_clock_path_fall

max_transition_clock_path_rise

max_transition_data_path_fall

max_transition_data_path_rise

A floating point attribute set with **set_max_transition**.

object_class

Type: string
A string attribute with the value of "clock". Each object class has a different object class value. The possible object classes are: design, cell, pin, port, net, lib, lib_cell, lib_pin, clock, scenario, input_delay, output_delay, timing_exception, timing_arc, lib_timing_arc.

period

Type: float
Represents the period of the clock.

propagated_clock

Type: Boolean
The value is "true" if this clock has propagated clock latency, "false" if it has ideal latency.

setup_uncertainty

Type: float

Represents the user-specified clock uncertainty for setup.

sources

Type: collection

waveform

Type: string
The value represents the clock waveform as a list of edges.

SEE ALSO

get_attribute(2)
list_attributes(2)
set_active_clocks(2)
set_clock_uncertainty(2)

clock_group_attributes

Lists the predefined attributes for clock_group objects.

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Attributes are properties assigned to objects. Definitions for clock_group attributes are provided in the subsections that follow.

Attributes are informational, or "read-only." You cannot set the value of attributes designed by the application.

To determine the value of an attribute, use the **get_attribute** command. To see a list of all attributes available for a class of objects use the **list_attributes -application** command.

Clock Group Attributes

clock_group_type

Type: string
Indicates the type of exception. Valid values are logically exclusive, asynchronous, and physically exclusive.

file_line_info

Type: string
A string attribute. If it exists, the value contains the Tcl or SDC source file name and line number pair of the command which was used to create this object.

full_name

Type: string
Returns a unique name for the clock_group. You create a unique name or the tool generates a name if you do not specify one.

group_count

Type: integer
Gives the number of individual groups given in the **set_clock_group** command.

is_allow_path

Type: Boolean
A Boolean attribute which is true if the -allow_paths option is used in the **set_clock_group** command for an asynchronous clock group setting.

object_class

Type: string
A string attribute, with the value "clock_group".

SEE ALSO

get_clock_groups(2)
get_attribute(2)
list_attributes(2)
report_clocks(2)
set_clock_groups(2)

clock_group_group_attributes

Lists the predefined attributes for clock_group_group objects.

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Attributes are properties assigned to objects. Definitions for clock_group attributes are provided in the subsections that follow.

Attributes are informational, or "read-only." You cannot set the value of attributes designed by the application.

To determine the value of an attribute, use the **get_attribute** command. To see a list of all attributes available for a class of objects use the **list_attributes -application** command.

Clock Group Group Attributes

object_class

Type: string
A string attribute, with the value "clock_group_group".

objects

Type: collection
The value is a collection of clocks contained in this group of the clock_group.

SEE ALSO

get_clock_groups(2)
get_clock_group_groups(2)
get_attribute(2)
list_attributes(2)
report_clocks(2)
set_clock_groups(2)

collection_deletion_effort

collection_deletion_effort

TYPE

string

DEFAULT

low

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

The **collection_deletion_effort** variable controls how saved collections are deleted when objects within them are potentially going out of scope. Allowed values are *low*, *medium*, or *high*, indicating to Galaxy Constraint Analyzer how much effort to expend to **preserve** a collection (or part of it) when objects are going out of scope.

Objects in a collection can go out of scope at several different times. When a cell is swapped or when the design is unlinked (that is, when another design is linked, causing the current linked design to become unlinked), a subset of objects in the design are removed, and collections can be affected. When the design/library which owns the objects is deleted, the collection is always deleted; **collection_deletion_effort** has no effect.

A collection is created relative to the current instance. That hierarchical node becomes the *root* of the collection. When hierarchical nodes that are being removed are above the root of a collection, the collection is always deleted. However, when the root of a collection is in the parent chain of the hierarchical node that is being deleted (henceforth referred to as the swap node), the collection is deleted based on the value of **collection_deletion_effort**:

- If the effort is *low*, the collection is deleted.
- If the effort is *medium*, the collection is deleted if **any** element of the collection has the swap node in its parent chain.
- If the effort is *high*, individual elements of the collection with the swap node in their parent chain are removed from the collection. The collection is deleted if it becomes empty.

The CPU cost increases from *low* to *high*. In most cases, *low* is a satisfactory choice.

To determine the current value of this variable, type **printvar collection_deletion_effort** or **echo $collection_deletion_effort**.

EXAMPLES

The following example illustrates the effects of using *low*, *medium*, or *high*.

In design 'M', two nodes i1 and i2 reference design 'I'. The following collections are created:

```
ptc_shell> set s1 [get_cells {i* i1/* i2/*}]
_sel27
ptc_shell> query_objects $s1
{"i1", "i2", "i1/low", "i1/low2", "i1/low3", "i2/low", "i2/low2", "i2/low3"}
ptc_shell> current_instance i1
i1
ptc_shell> set s2 [get_cells *]
_sel28
ptc_shell> query_objects $s2
{"i1/low", "i1/low2", "i1/low3"}
ptc_shell> current_instance
Current instance is the top-level of design 'M'.
```

With **collection_deletion_effort** at *low*, you can swap i2/low with no effect on collection _sel28, stored in variable s2, because the root was i1. However, the collection _sel27, stored in variable s1, is deleted because its root (the top of design M) is in the parent chain of the swap node (i2/low).

With collection_deletion_effort at *medium*, you can swap i2/low with no effect on either collection _sel27 or _sel28. Here, the swap node i2/low is not above any element of either collection.

With collection_deletion_effort at *high*, you can swap i2 with no effect on collection _sel28. For collection _sel27, the elements in the collection with i2 in their parent chain are removed, leaving the following:

```
ptc_shell> query_objects $s1
{"i1", "i2", "i1/low", "i1/low2", "i1/low3"}
```

SEE ALSO

collections(2)
link_design(2)
printvar(2)
swap_cell(2)

collection_result_display_limit

Sets the maximum number of objects that can be displayed by any command that displays a collection.

TYPE

int

DEFAULT

100

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

This variable sets the maximum number of objects that can be displayed by any command that displays a collection. The default is 100.

When a command (for example, **add_to_collection**) is issued at the command prompt, its result is implicitly queried, as though **query_objects** had been called. You can limit the number of objects displayed by setting this variable to an appropriate integer. A value of -1 displays all objects; a value of 0 displays the collection handle id instead of the names of any objects in the collection.

To determine the current value of this variable, use **printvar collection_result_display_limit**.

SEE ALSO

collections(2)
printvar(2)
query_objects(2)

compare_clock_debug_max_pin

Specifies the maximum number of clock pins that the GUI displays in the Info Pane for a clock violation in the compare_block_to_top output.

TYPE

int

DEFAULT

50

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

These clock pins are the ones at which the clock violation was detected.

SEE ALSO

compare_block_to_top(2)

compare_clock_tolerance_ps

Specifies the tolerance value (in ps) while comparing two clock waveforms in the B2T_CLK_0003 rule.

TYPE

Double

DEFAULT

1 ps

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Specifies the tolerance value while comparing two clock waveforms for the B2T_CLK_0003 rule. If the difference of two clock waveforms is less than the tolerance value specified, a B2T_CLK_0003 rule violation is not reported.

To determine the current value of this variable, type **printvar compare_clock_tolerance_ps** or **echo $compaer_clock_tolerance_ps**.

SEE ALSO

set_clock_transition(2)

compare_exception_debug_max_path

Specifies the maximum number of paths that the GUI displays in the Info Pane for an exception violation in the compare_block_to_top output.

TYPE

int

DEFAULT

50

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

These paths are the ones at which the exception violation was detected.

SEE ALSO

compare_block_to_top(2)

compare_unpropagated_clocks

Enables comparison of unpropagated clocks in s2s scenario.

TYPE

Boolean

DEFAULT

false

GROUP

Timing variables

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

When this variable is set to *true*, the tool also flags clock differences for the clocks which do not drive a sequential element. In the current behavior, the tool does not flag clock differences if the clocks do not drive a sequential element.

To determine the current value of this variable, type **printvar compare_unpropagated_clocks** or **echo $compare_unpropagated_clocks**.

SEE ALSO

timing_remove_clock_reconvergence_pessimism(2)

constraints_compare_full_behavior

Specifies whether case analysis and disabled timing arcs are also compared using the behavioral checker in the **compare_constraints** command, in addition to the default behavior.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This variable is available only if you invoke the pt_shell with the **-constraints** option.

When *false* (the default), **compare_constraints** command compares case analysis and disabled timing arcs at the definition points. If the definition points are unmapped using the name mapping file (or **define_name_maps** command), it will throw violations for missing or mismatching case / disabled arcs. Clocks and exceptions are compared using the behavioral checker.

When the **constraints_compare_full_behavior** variable is *true*, even if the case analysis and disabled arcs that have non-mapped definition points, but show the same behavior at timing path endpoint both scenarios / designs, will not be shown as violations. Basically, case/disable arcs that are not defined at the same definition points, but have the same impact at the timing path endpoints, will not be flagged as violations.

To determine the current value of this variable, use **printvar constraints_compare_full_behavior**.

SEE ALSO

compare_constraints(2)
report_constraint_analysis(2)
define_name_maps(2)

create_clock_no_input_delay

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Affects delay propagation characteristics of clock sources created using **create_clock**. When *false* (the default), clock sources used in the data path are established as timing startpoints. The clock sources in the design propagate rising delays on every rising clock edge, and propagate falling delays on every falling clock edge. Disable this behavior by setting **create_clock_no_input_delay** to *true*.

To determine the current value of this variable, use **printvar create_clock_no_input_delay**.

SEE ALSO

create_clock(2)

dc_synopsys_root

Specifies the installation root containing DesignCompiler.

TYPE

string

DEFAULT

none

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

This variable can be set to give a location of the DesignCompiler installation root. This location checks that the version of Design Compiler supports the constraint commands given to constraint consistency.

To determine the current value of this variable, use the **printvar dc_synopsys_root** command.

For example:

```
set_app_var dc_synopsys_root /u/release/synthesis/M-2016.12
```

The **analyze_design** command can perform rule checks for whether commands are compatible across the multiple products. These are rules CMP_0001 for commands and CMP_0002 for options of commands.

SEE ALSO

icc_synopsys_root(3)
pt_synopsys_root(3)
analyze_design(2)

design_attributes

Describes the predefined attributes for designs.

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Attributes are properties assigned to objects such as pins, cells and nets. Definitions for design attributes are provided in the subsections that follow.

Attributes are informational, or "read-only." You cannot set the value of attributes designed by the application.

To determine the value of an attribute, use the **get_attribute** command. To see a list of all attributes available for a class of objects use the **list_attributes -application** command.

Design Attributes

full_name

A string attribute for the name of a design.

is_current

A Boolean attribute. The value is "true" if this design is the current_design for the session, "false" otherwise.

max_capacitance

A floating point attribute set with **set_max_capacitance**.

max_fanout

A floating point attribute set with **set_max_fanout**.

max_transition

A floating point attribute set with **set_max_transition**.

object_class

A string attribute with the value of "design". Each object class has a different object class value. The possible object classes are: design, cell, pin, port, net, lib, lib_cell, lib_pin, clock, scenario, input_delay, output_delay, timing_exception, timing_arc, lib_timing_arc.

operating_condition_max

A string attribute that specifies the name of maximum operating condition.

operating_condition_min

A string attribute that specifies the name of minimum operating condition.

process_max

A float attribute that specifies the process scaling factor for the maximum operating condition. Allowed values are 0.0 through 100.0.

process_min

A float attribute that specifies the process scaling factor for the minimum operating condition. Allowed values are 0.0 through 100.0.

temperature_max

A float attribute that specifies the temperature value, in degrees Celsius, for the maximum operating condition. Allowed values are -300.0 through +500.0.

temperature_min

A float attribute that specifies the temperature value, in degrees Celsius, for the minimum operating condition. Allowed values are -300.0 through +500.0.

tree_type_max

A string attribute that specifies the tree type for the maximum operating condition. Allowed values are best_case_tree, balanced_tree, or worst_case_tree. The tree type is used to estimate interconnect delays by providing a model of the RC tree.

tree_type_min

A string attribute that specifies the tree type for the minimum operating condition. Allowed values are best_case_tree, balanced_tree, or worst_case_tree. The tree type is used to estimate interconnect delays by providing a model of the RC tree.

voltage_max

A float attribute that specifies the voltage value, in volts, for the maximum operating condition. Allowed values are 0.0 through 1000.0.

voltage_min

A float attribute that specifies the voltage value, in volts, for the minimum operating condition. Allowed values are 0.0 through 1000.0.

SEE ALSO

current_design(2)
get_attribute(2)
list_attributes(2)

disable_case_analysis

Specifies whether case analysis is disabled.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

When *false* (the default), constant propagation is performed in the design from pins either that are tied to a logic constant value, or for which a **case_analysis** command is specified. For example, a typical design has several pins set to a constant logic value. By default, this constant value propagates through the logic to which it connects. When the variable **disable_case_analysis** is *true*, case analysis and constant propagation are not performed.

To determine the current value of this variable, use **printvar disable_case_analysis**.

If the variable **disable_case_analysis_ti_hi_lo** is set to *true* then constant propagation from pins that are tied to a logic constant value will not be performed.

SEE ALSO

disable_case_analysis_ti_hi_lo(3)
remove_case_analysis(2)
report_case_analysis(2)
set_case_analysis(2)

disable_case_analysis_ti_hi_lo

Specifies if logic constants should be propagated from pins that are tied to a logic constant value.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

When *false* (the default), constant propagation is performed from pins that are tied to a logic constant value.

For example, a typical design has several pins set to a constant logic value. By default, this constant value propagates through the logic to which it connects. When the **disable_case_analysis_ti_hi_lo** variable is *true*, constant propagation is not performed from these pins.

This current value of this variable does not alter the propagation of logic values from pins where the logic value has been set by the **set_case_analsyis** command.

To determine the current value of this variable, use **printvar disable_case_analysis_ti_hi_lo**.

If the variable **disable_case_analysis** is set to *true* then all constant propagation is disabled regardless of the current value of the **disable_case_analysis_ti_hi_lo** variable.

SEE ALSO

disable_case_analysis(3)
remove_case_analysis(2)
report_case_analysis(2)
set_case_analysis(2)

display_violations_per_rule_limit

Specifies the maximum number of violations that are displayed per rule, within one scenario or within the scenario-independent netlist checks.

TYPE

integer

DEFAULT

10000000

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

If there are more than the number of violations, the total count is displayed but details are shown for only the specified number of violations.

SEE ALSO

analyze_design(2)

display_waived_violations_per_rule_limit

Specifies the maximum number of waived violations that will be displayed per rule, within one scenario or within the scenario-independent netlist checks.

TYPE

int

DEFAULT

10000000

DESCRIPTION

If there are more than the number of waived violations, the total count will be displayed. Details are shown for only the specified number of violations.

SEE ALSO

analyze_design(2)

enable_clock_attribute_on_hier_pins

Enables clock attributes to be returned for hierarchical pins.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

When *false* (the default), if a hierarchical pin lies on a clock network and you query for the 'clocks' attribute on the pin with the **get_attribute** command, the tool reports that no clocks exist on the hier pin (ATTR-3).

When *true*, if a hierarchical pin lies on a clock network and you query for the 'clocks' attribute on the pin with the **get_attribute** command, the tool returns the list of clocks that exist on the hier pin.

SEE ALSO

get_attribute(2)

exception_attributes

Lists the predefined attributes for exception objects.

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Attributes are properties assigned to objects. Definitions for exception attributes are provided in the subsections that follow.

Attributes are informational, or "read-only." You cannot set the value of attributes designed by the application.

To determine the value of an attribute, use the **get_attribute** command. To see a list of all attributes available for a class of objects use the **list_attributes -application** command.

Exception Attributes

command_text

A string attribute. The value is the equivalent commands that can re-create the object.

exception_type

A string attribute indicating the type of exception. Valid values are "false_path", "multicycle_path", and "max_min_delay".

file_line_info

A string attribute. If it exists, the value contains one or more Tcl or SDC source file name and line number pairs of the commands which were used to create this object.

full_name

A string attribute returning a unique name for the exception.

group_count

An int attribute specifying the number of groups in the exception.

has_from

A Boolean attribute; true if the exception has a -from group.

has_through

A Boolean attribute; true if the exception has a -through group.

has_to

A Boolean attribute; true if the exception has a -to group.

hold_fall_end

A Boolean attribute; true if the exception is multicycle and the hold fall multiplier is relative to the end clock.

hold_fall_multiplier

An int attribute for multicycle_path exceptions, specifying the multiplier value for hold fall.

hold_rise_end

A Boolean attribute; true if the exception is multicycle and the hold rise multiplier is relative to the end clock.

hold_rise_multiplier

An int attribute for multicycle_path exceptions, specifying the multiplier value for hold rise.

is_hold_fall_false

A Boolean attribute; true if the exception is false_path, and the hold fall timing is set to be false.

is_hold_rise_false

A Boolean attribute; true if the exception is false_path, and the hold rise timing is set to be false.

is_setup_fall_false

A Boolean attribute; true if the exception is false_path, and the setup fall timing is set to be false.

is_setup_rise_false

A Boolean attribute; true if the exception is false_path, and the setup rise timing is set to be false.

max_fall_delay

A float attribute for max_min_delay exceptions. Specifies the delay value for max_fall.

max_rise_delay

A float attribute for max_min_delay exceptions. Specifies the delay value for max_rise.

min_fall_delay

A float attribute for max_min_delay exceptions. Specifies the delay value for min_fall.

min_rise_delay

A float attribute for max_min_delay exceptions. Specifies the delay value for min_rise.

object_class

A string attribute, with the value "exception".

setup_fall_multiplier

An int attribute for multicycle_path exceptions, specifying the multiplier value for setup fall.

setup_fall_start

A Boolean attribute; true if the exception is multicycle and the setup fall multiplier is relative to the start clock.

setup_rise_multiplier

An int attribute for multicycle_path exceptions, specifying the multiplier value for setup rise.

setup_rise_start

A Boolean attribute; true if the exception is multicycle and the setup rise multiplier is relative to the start clock.

through_group_count

An int attribute specifying the number of -through, -rise_through, and -fall_through options that were specified with the exception command. For example, if the exception was specified as "-through {a1 a2 a3}-rise_through {b1 b2 b3}", the **through_group_count** attribute value is 2.

SEE ALSO

get_exceptions(2)

get_attribute(2)
list_attributes(2)
report_exceptions(2)
set_false_path(2)
set_max_delay(2)
set_min_delay(2)
set_multicycle_path(2)

exception_group_attributes

Lists the predefined attributes for exception_group objects.

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Attributes are properties assigned to objects. Definitions for exception_group attributes are provided in the subsections that follow.

Attributes are informational, or "read-only." You cannot set the value of attributes designed by the application.

To determine the value of an attribute, use the **get_attribute** command. To see a list of all attributes available for a class of objects use the **list_attributes -application** command.

Exception Group Attributes

edge

A string attribute, gives the edge type of the from/through/to group. It can be one of "rise", "fall", or "both.

exception_object

A collection type attribute containing exactly one exception. This exception is the one which contains this exception_object.

full_name

A string attribute returning the type of from/through/to group. It can return one of "from", "rise_from", "fall_from", ... "rise_to" or "fall_to".

has_clock

A Boolean attribute, which is true if the this from/through/to group contains a clock.

object_class

A string attribute, with the value "exception_group".

objects

A collection attribute containing heterogeneous objects of the type pins, ports and clocks. This returns all the objects which are in this exception_group object.

type

A string attribute returning the type of group. It can return one of "from", "to" or "through".

SEE ALSO

get_exceptions(2)
get_exception_groups(2)
get_attribute(2)
list_attributes(2)

filter_collection_extended_syntax

TYPE

boolean

DEFAULT

application specific

DESCRIPTION

This variable controls whether the filter_collection command supports extended math expressions. Please see the man page for filter_collection for details.

SEE ALSO

filter_collection(2)

gca_tmp_dir

Specifies the directory that constraint consistency uses for temporary storage.

TYPE

string

DEFAULT

/tmp (the local /tmp partition)

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Specifies a directory for the tool to use as temporary storage. By default, the value is "/tmp". You can set this variable to any directory with proper read/write permissions, such as ".", the current directory.

To determine the current value of this variable, type **printvar gca_tmp_dir** or **echo $gca_tmp_dir**

SEE ALSO

set_program_options(1)

get_timing_arcs_include_arcs_from_to_hier_pins

Controls the behavior of the **get_timing_arcs -of_objects** command on a net.

TYPE

Boolean

DEFAULT

true

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

This variable controls whether the **get_timing_arcs -of_objects** *net* command includes arcs from or to hierarchical pins. When certain constraints (such as create_clock) are specified on hierarchical pins, those pins become path startpoints or endpoints, and the timing arcs of the net are modified. If this variable is true, the modified arcs from or to such hierarchical pins are included in the result of the **get_timing_paths** command when the **-of_objects** option is used on such a net.

For the current value of this variable, type **get_timing_arcs_include_arcs_from_to_hier_pins**.

SEE ALSO

printvar(2)
get_timing_arcs(2)

grouping_violations_hierarchy_separator_limit

Specifies how many hierarchical separators can be considered in the names of netlist objects while performing violation grouping for violation browser.

TYPE

integer

DEFAULT

1000000

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Specifies how many hierarchical separators can be considered in the names of netlist objects while performing violation grouping for violation browser. The number of hierarchical separators considered are counted from leaf level to the top level.

By default, the value of the variable is very high which means that all levels of hierarchy are considered for grouping. You can set this variable to 1 to only considered grouping at the leaf level. You can set this variable to 0 to disable all violation grouping.

Note that violation grouping groups violations that are only different in numerical portions of the netlist object names that occur in the parameters of the violation. Also, note that grouping only affects GUI output in violation browser and does not affect text output.

Note the hierarchical separator is specified via the **hierarchy_separator** variable.

SEE ALSO

hierarchy_separator(3)

gui_annotation_attributes

Description of the predefined attributes for gui_annotation.

DESCRIPTION

Objects of type gui_annotation are created by the **gui_add_annotation** command. See the manpage for that command for more details.

To determine the value of an attribute use the **get_attribute** command.

GUI Annotation Attributes

client_data

A string attribute that can be used by client code to store arbitrary information

color

The color used to draw the annotation.

fill_pattern

The fill pattern used to fill the annotationshape

group

The group of the annotation.

info_tip

The tip text (or command) displayed when a user hovers over the annotation in the layout.

line_style

The line style to use when drawing the annotaiton

line_width

The width of the line in pixels.

object_class

The string attribute with the value "gui_annotation"

points

The points for the annotation

query_command

The command to use when the annotation is queried in the layout.

query_text

The text to display when the annotation is queried in the layout.

shape_type

The type of shape for this annotation. For example, rect, line, etc.

text

If the shape_type is text this attribute has the text to display

window

When set this annotation will only be drawn on the specified window.

SEE ALSO

gui_create_annotation(2)
gui_get_annotations(2)
gui_remove_all_annotations(2)
gui_remove_annotation(2)

gui_build_query_data_table

Controls whether the tool initializes data for fast data query after the GUI starts.

TYPE

Boolean

DEFAULT

The default value for **gui_build_query_data_table** is true.

DESCRIPTION

This variable controls whether IC Compiler initializes query data during GUI startup. By default, the **gui_build_query_data_table** variable is true and IC Compiler initializes query data, which might require some time for a large design. You can set the variable to false to prevent the tool from initializing query data.

EXAMPLES

The following example prevents the tool from initializing query data.

```
prompt> set gui_build_query_data_table false
```

SEE ALSO

printvar(2)

gui_custom_setup_files

Variable for specifying GUI customization files to be loaded during GUI startup.

TYPE

Tcl list of fully-qualified file names

DEFAULT

$synopsys/admin/setup/.synopsys_<app>_gui.tcl

DESCRIPTION

This variable specifies a set of files that should be sourced when the GUI starts up. You can add this variable setting to the application setup file to provide GUI customizations to individuals, or share customizations among a group of users. A GUI customization file typically contains commands to specify hotkeys, menus, or toolbars which implement specific functions to support a custom environment or flow.

The GUI initializes and searches the list of files in order. Each file in the list is sourced. Finally, your .synopsys_<app>_gui.tcl file is loaded to complete the customizations.

You can disable the loading of the customizations by setting the **gui_disable_custom_setup** variable.

SEE ALSO

printvar(2)
gui_disable_custom_setup(3)

gui_default_window_type

,IP **gui_default_window_type** Read-only variable specifying the default window type for GUI customization commands.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This read-only variable contains the name of the window type that is used as the default when a menu, hotkey, or toolbar customization is specified without specifying a window type explicitly.

SEE ALSO

gui_create_menu(2)
gui_create_toolbar(2)
gui_create_toolbar_item(2)
gui_delete_menu(2)
gui_delete_toolbar(2)
gui_delete_toolbar_item(2)
gui_report_hotkeys(2)
gui_set_hotkey(2)
printvar(2)

gui_disable_custom_setup

Variable for disabling gui customizations

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This variable specifies whether GUI customization loading is disabled during GUI startup. Set the variable to true to prevent GUI customizations from loading. This includes customizations specified in the **gui_custom_setup_files** variable as well as your .synopsys_<app>_gui.tcl file.

SEE ALSO

printvar(2)
gui_custom_setup_files(3)

gui_object_attributes

Description of the predefined attributes for gui_object.

DESCRIPTION

Objects of type gui_object are created by the **gui_create_vm_objects** command. See the manpage for that command for more details.

All attributes are read-only. To determine the value of an attribute use the **get_attribute** command.

GUI Object Attributes

name

A string attribute for the name of the object

full_name

The same as name

object_class

A string attribute with the value of "gui_object"

layout_info_tip

The info-tip to use in the layout when the user hovers over this object.

query_text

The query text to display if the user queries this object in the layout.

core_object

A collection containing the actual object wrapped by this gui_object.

SEE ALSO

gui_create_vm_objects(2)
gui_create_vm(2)
gui_create_vm_bucket(2)
gui_update_vm_annotations(2)

gui_online_browser

Specifies the name of the browser used to invoke the online help system from the help menu of the product.

TYPE

string

DEFAULT

netscape

GROUP

gui

DESCRIPTION

This string variable holds the value of the default browser which is used to invoke online help system from Help menu of the application. The value the variable should be either **netscape**, **mozilla** or **firefox**.

If you specify a browser other than those listed above, the application defaults to the netscape browser.

Use the following command to determine the current value of the variable:

```
prompt> printvar gui_online_browser
```

SEE ALSO

gui_custom_setup_files(3)

gui_suppress_auto_layout

Variable for disabling automatic opening of the Layout window.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

Specifies whether the Layout window is prevented from opening when loading a new design.

SEE ALSO

printvar(2)

hide_waived_violations

Specifies whether waived violations are hidden from the GUI and constraint analysis report.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

When *false* (the default), all constraint violations are shown in the report of report_constraint_analysis -include {violations}. That includes the violations that are waived by the violation waivers. When the **hide_waived_violations** variable is *true*, the waived violations are hidden from the constraint analysis report. The report shows only the violations that are not waived by the waivers.

The **hide_waived_violations** variable is associated with the **Hide Waived Violations** toggle button on the toolbar in the constraint consistency GUI. If the variable is set to *false*, the GUI violations browsers show all of the violations including the waived ones; when the variable is set to *true*, the violation browsers show only the unwaived violations.

To determine the current value of this variable, use **printvar hide_waived_violations**. Its value controls both reporting and GUI behavior. The GUI is dynamically updated (to remove the waivers) if this variable is set to *true* and violations are waived during a debug session in the GUI.

SEE ALSO

report_waiver(2)
create_waiver(2)
report_constraint_analysis(2)

hierarchy_separator

Determines how hierarchical elements of the netlist are delimited in reports, and searched for in selections and other commands.

TYPE

string

DEFAULT

/ (forward slash)

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

By default, the value is the slash, "/". The choice of a separator is limited to these characters: bar "|", caret "^", at "@", dot ".", and slash "/".

Normally, you should accept the default slash. However, in some cases where the hierarchy character is embedded in some names, the search engine might produce results that are not intended; the **hierarchy_separator** is a convenient method for dealing with this situation. For example, consider a design that contains a hierarchical cell A, which contains hierarchical cells B and B/C; B/C contains D; B contains C. Searching for "A/B/C/D" is ambiguous and might not match what you intended. However, if you set the **hierarchy_separator** to the vertical bar "|", searching for "A | B/C | D" is very explicit, as is "A | B | C | D".

SEE ALSO

selection(2)

icc_synopsys_root

Specifies installation root containing IC Compiler.

TYPE

String

DEFAULT

none

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

This variable can be set to give a location of the IC Compiler installation root. This location is used to check that the version of IC Compiler supports the constraint commands given to constraint consistency.

To determine the current value of this variable, use **printvar icc_synopsys_root**.

For example:

```
set_app_var icc_synopsys_root /u/release/icc/M-2016.12
```

The command **analyze_design** can perform rule checks for whether commands are compatible across the multiple products. These are rules CMP_0001 for commands and CMP_0002 for options of commands.

SEE ALSO

icc_synopsys_root(3)
pt_synopsys_root(3)
analyze_design(2)

in_gui_session

This read-only variable is "true" when the GUI is active and "false" when the GUI is not active.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This variable can be used in writing Tcl code that depends on the presence the graphical user interface (GUI). The read-only variable has the value "true" if **gui_start** has been invoked and the GUI is active. Otherwise, the variable has the value "false" (default).

SEE ALSO

printvar(2)
gui_start(2)
gui_stop(2)

input_delay_attributes

Describes the predefined attributes for input delay objects.

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Attributes are properties assigned to objects such as pins, cells and nets. Definitions for input_delay attributes are provided in the subsections that follow.

Attributes are informational, or "read-only." You cannot set the value of attributes designed by the application.

To determine the value of an attribute, use the **get_attribute** command. To see a list of all attributes available for a class of objects use the **list_attributes -application** command.

Input_Delay Attributes

clock_name

A string attribute representing the name of the relative clock.

file_line_info

A string attribute. If it exists, the value contains one or more Tcl or SDC source file name and line number pairs of the commands which were used to create this object.

full_name

A string attribute representing a unique name for the input delay object.

is_clock_fall

A Boolean attribute. The value is "true" if this input_delay is relative to the falling edge of the clock, "false" otherwise.

is_level_sensitive

A Boolean attribute. The value is "true" if this input_delay represents a level-sensitive startpoint, "false" otherwise.

is_network_latency_included

A Boolean attribute. The value is "true" if this input_delay includes clock network latency, "false" otherwise.

is_source_latency_included

A Boolean attribute. The value is "true" if this input_delay includes clock source latency, "false" otherwise.

max_fall

A float attribute representing the maximum fall delay value.

max_rise

A float attribute representing the maximum rise delay value.

min_fall

A float attribute representing the minimum fall delay value.

min_rise

A float attribute representing the minimum rise delay value.

object_class

A string attribute with the value of "input_delay". Each object class has a different object class value.

SEE ALSO

set_input_delay(2)
get_attribute(2)
list_attributes(2)

lib_attributes

Lists the predefined attributes for library objects.

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Attributes are properties assigned to objects such as lib_pins, pins, cells and nets. Definitions for lib attributes are provided in the subsections that follow.

Attributes are informational, or "read-only." You cannot set the value of attributes designed by the application.

To determine the value of an attribute, use the **get_attribute** command. To see a list of all attributes available for a class of objects use the **list_attributes -application** command.

Library Attributes

capacitance_unit

A string attribute representing the capacitance unit for this library.

current_unit

A string attribute representing the current unit for this library.

default_fanout_load

A floating point attribute representing the default_fanout_load value of this library.

default_max_capacitance

A floating point attribute representing the default_max_capacitance value of this library.

default_max_fanout

A floating point attribute representing the default_max_fanout value of this library.

default_max_transition

A floating point attribute representing the default_max_transition value of this library.

extended_name

A string attribute representing the full path to the source .db file of this library plus the library name, separated by a ':' character. For example, "/disks/my_dir/work/tech_lib.db:tech_lib".

full_name

A string attribute for the name of a library.

object_class

A string attribute with the value of "lib". Each object class has a different object class value.

resistance_unit

A string attribute representing the resistance unit for this library.

source_file_name

A string attribute representing the full path to the source .db file of this library.

temperature_unit

A string attribute representing the temperature unit for this library.

time_unit

A string attribute representing the time unit for this library.

voltage_unit

A string attribute representing the voltage unit for this library.

SEE ALSO

find(2)
get_attribute(2)
list_attributes(2)

lib_cell_attributes

Lists the predefined attributes for lib_cells.

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Attributes are properties assigned to objects such as lib_pins, pins, cells and nets. Definitions for lib_cell attributes are provided in the subsections that follow.

Attributes are informational, or "read-only." You cannot set the value of attributes designed by the application.

To determine the value of an attribute, use the **get_attribute** command. To see a list of all attributes available for a class of objects use the **list_attributes -application** command.

Library Cell Attributes

area

A float attribute for the area of a lib_cell.

base_name

A string attribute for the name of a lib_cell. This name does not include a library prefix.

disable_timing

A Boolean attribute that is true if the lib_cell is marked as disable_timing in the library.

dont_touch

A Boolean attribute that is true if the lib_cell is marked as dont_touch to prevent optimization from changing any instances of this lib_cell.

dont_use

A Boolean attribute that is true if the lib_cell is marked as dont_use to prevent optimization from inserting instances of this lib_cell.

extended_name

A string attribute representing the full path to the source .db file of the library containing this lib_cell plus the full name of the lib_cell, separated by a ':' character. For example, "/disks/my_dir/work/tech_lib.db:tech_lib/AN2".

full_name

A string attribute for the name of a lib_cell. This name includes a prefix for the library of the lib_cell.

function_id

A string attribute representing the logic function of the lib_cell.

is_black_box

A Boolean attribute. The value is "true" if this lib_cell has no logic function.

is_combinational

A Boolean attribute that is true if the lib_cell is not sequential.

is_fall_edge_triggered

A Boolean attribute that is true if the lib_cell has timing behavior relative to the falling edge of a clock signal.

is_integrated_clock_gating_cell

A Boolean attribute that is true if the lib_cell is defined in the library as an integrated clock gating cell.

is_memory_cell

A Boolean attribute that is true if the lib_cell is defined in the library as a memory cell.

is_mux

A Boolean attribute that is true if the lib cell is a mux.

is_negative_level_sensitive

A Boolean attribute that is true if the lib_cell has negative level-sensitive timing behavior, as in a negative D-latch.

is_pad_cell

A Boolean attribute that is true if the lib_cell is defined in the library as a pad cell.

is_positive_level_sensitive

A Boolean attribute that is true if the lib_cell has positive level-sensitive timing behavior, as in a positive D-latch.

is_rise_edge_triggered

A Boolean attribute that is true if the lib_cell has timing behavior relative to the rising edge of a clock signal.

is_sequential

A Boolean attribute that is true if the lib_cell has sequential logic function.

is_three_state

A Boolean attribute that is true if the lib_cell has one or more three_state outputs.

number_of_pins

An integer attribute representing the number of pins on this lib_cell.

object_class

A string attribute with the value of "lib_cell". Each object class has a different object class value.

SEE ALSO

find(2)
get_attribute(2)
list_attributes(2)

lib_pin_attributes

Lists the predefined attributes for lib_pins.

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Attributes are properties assigned to objects such as lib_pins, pins, cells and nets. Definitions for lib_pin attributes are provided in the subsections that follow.

Attributes are informational, or "read-only." You cannot set the value of attributes designed by the application.

To determine the value of an attribute, use the **get_attribute** command. To see a list of all attributes available for a class of objects use the **list_attributes -application** command.

Library Pin Attributes

base_name

A string attribute for the name of a lib_pin. This name does not include a library, lib_cell prefix.

clock

A Boolean attribute that is true if the lib_pin is defined as a clock in the library.

direction

A string attribute for the direction of a pin. Possible values are *in*, *out*, *inout*, or *unknown*.

disable_timing

A Boolean attribute that is true if the lib_pin is marked as disable_timing in the library.

extended_name

A string attribute representing the full path to the source .db file of the library containing this lib_pin plus the full name of the lib_pin, separated by a ':' character. For example, "/disks/my_dir/work/tech_lib.db:tech_lib/AN2/Z".

full_name

A string attribute for the name of a lib_pin. This name includes a prefix for the library and lib_cell of the lib_pin.

function

A string attribute on output or inout lib_pins for the logic function. The logic function is defined in the library.

is_async_pin

A Boolean attribute that is true if this lib_pin is a preset or clear pin.

is_clear_pin

A Boolean attribute that is true if this lib_pin is a clear (reset) pin.

is_clock_pin

A Boolean attribute that is true for clock pins of registers. A clock pin is any pin that has a timing check from it to a data pin.

is_data_pin

A Boolean attribute that is true for data pins of registers. A data pin is any pin that has a timing check to it.

is_fall_edge_triggered_clock_pin

A Boolean attribute that is true for clock pins of registers with falling edge-triggered behavior.

is_fall_edge_triggered_data_pin

A Boolean attribute that is true for data pins of registers with falling edge-triggered behavior.

is_mux_select_pin

A Boolean attribute that is true if the lib pin is a select pin of a mux lib cell.

is_negative_level_sensitive_clock_pin

A Boolean attribute that is true for clock pins of latches with negative level-sensitive behavior.

is_negative_level_sensitive_data_pin

A Boolean attribute that is true for data pins of latches with negative level-sensitive behavior.

is_positive_level_sensitive_clock_pin

A Boolean attribute that is true for clock pins of latches with positive level-sensitive behavior.

is_positive_level_sensitive_data_pin

A Boolean attribute that is true for data pins of latches with positive level-sensitive behavior.

is_preset_pin

A Boolean attribute that is true if this lib_pin is an asynchronous preset pin.

is_rise_edge_triggered_clock_pin

A Boolean attribute that is true for clock pins of registers with rising edge-triggered behavior.

is_rise_edge_triggered_data_pin

A Boolean attribute that is true for data pins of registers with rising edge-triggered behavior.

is_three_state

A Boolean attribute. Possible values are "true" and "false". A library pin is a three-state pin if there arethree-state enable and or three-state disable arcs from or to that pin.

is_three_state_enable_pin

A Boolean attribute. Possible values are "true" and "false". A library pin is a three-state enable pin if there arethree-state enable and or three-state disable arcs from that pin.

is_three_state_output_pin

A Boolean attribute. Possible values are "true" and "false". A library pin is a three-state output pin if there arethree-state enable and or three-state disable arcs to that pin.

max_capacitance

A floating point attribute representing the max_capacitance design rule of this lib_pin.

max_fanout

A floating point attribute representing the max_fanout design rule of this lib_pin.

max_transition

A floating point attribute representing the max_transition design rule of this lib_pin.

object_class

A string attribute with the value of "lib_pin". Each object class has a different object class value.

pin_capacitance

A float attribute representing the capacitance of this lib_pin.

pin_direction

A string attribute for the direction of a pin. Possible values are *in*, *out*, *inout*, or *unknown*. This is the same as "direction". This variable is supported for compatibility with other tools. The "direction" attribute is supported for both ports and pins and therefore easier to use for heterogeneous collections of pins and ports.

three_state_function

A string attribute on output or inout lib_pins for the three_state logic function. The three_state logic function is defined in the library. If the three_state logic function evaluates to zero, then the lib_pin is enabled.

SEE ALSO

find(2)
get_attribute(2)
list_attributes(2)

lib_timing_arc_attributes

man page to document the available lib_timing_arc attributes.

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Attributes are properties assigned to objects such as timing arcs, pins, cells and nets. Definitions for lib_timing_arc attributes are provided in the subsections that follow.

Attributes are informational, or "read-only." You cannot set the value of attributes designed by the application.

To determine the value of an attribute, use the **get_attribute** command. To see a list of all attributes available for a class of objects use the **list_attributes -application** command.

Collections of library timing arcs are created with the command **get_lib_timing_arcs**.

Library Timing Arc Attributes

from_lib_pin

A collection attribute returning the from library pin for a lib_timing_arc. If this lib_timing_arc is for a timing check such as setup or hold, the from pin is the clock pin of the check.

is_disabled

A boolean attribute for whether this library timing arc is disabled. Library timing arcs may be disable either because the user has disabled it with the command **set_disable_timing** or because of mode settings.

is_user_disabled

A boolean attribute for whether this library timing arc is disabled because of a **set_disable_timing** command.

mode

A string attribute with the name of the mode this lib timing arc is active for. The attribute will is only defined for moded arcs.

object_class

A string attribute with the value of "lib_timing_arc". Each object class has a different object class value. The possible object classes are: design, cell, pin, port, net, lib, lib_cell, lib_pin, clock, scenario, input_delay, output_delay, timing_exception, timing_arc, lib_timing_arc.

sense

A string attribute for the sense of a lib_timing_arc. Possible values are

rising_edge rising_to_rise falling_to_rise
falling_edge rising_to_fall falling_to_fall
rise_to_rise fall_to_rise rise_to_fall
fall_to_fall positive_unate negative_unate
non_unate clear_high clear_low
clear_both preset_high preset_low
preset_both disable_high disable_low
disable_both disable_high_rise
disable_low_rise disable_both_rise
disable_high_fall disable_low_fall
disable_both_fall
enable_high enable_low enable_both
enable_high_rise enable_low_rise enable_both_rise

enable_high_fall enable_low_fall enable_both_fall
retain_rising_edge retain_rise_to_rise
retain_fall_to_rise retain_falling_edge
retain_rise_to_fall retain_fall_to_fall
retain_positive_unate retain_negative_unate
retain
max_clock_tree_positive_unate max_clock_tree_rise_to_rise
max_clock_tree_fall_to_fall max_clock_tree_negative_unate
max_clock_tree_rise_to_fall max_clock_tree_fall_to_rise
max_clock_tree_non_unate min_clock_tree_positive_unate
min_clock_tree_rise_to_rise min_clock_tree_fall_to_fall
min_clock_tree_negative_unate min_clock_tree_rise_to_fall
min_clock_tree_fall_to_rise min_clock_tree_non_unate
nonseq_setup_clk_rise nonseq_setup_rise_clk_rise
nonseq_setup_fall_clk_rise nonseq_setup_clk_fall
nonseq_setup_rise_clk_fall nonseq_setup_fall_clk_fall
nonseq_hold_clk_rise nonseq_hold_rise_clk_rise
nonseq_hold_fall_clk_rise nonseq_hold_clk_fall
nonseq_hold_rise_clk_fall nonseq_hold_fall_clk_fall
clock_gating_setup_clk_rise clock_gating_setup_rise_clk_rise
clock_gating_setup_fall_clk_rise clock_gating_setup_clk_fall
clock_gating_setup_rise_clk_fall clock_gating_setup_fall_clk_fall
clock_gating_hold_clk_rise clock_gating_hold_rise_clk_rise

to_lib_pin

A collection attribute returning the to library pin for a lib_timing_arc. If this lib_timing_arc is for a timing check such as setup or hold, the from pin is the data pin of the check.

when

A string attribute returning the when value for the lib_timing_arc.

SEE ALSO

get_attribute(2)
list_attributes(2)

link_allow_design_mismatch

Controls the behavior of the link design when pin mismatch between instance and reference occur.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

This variable controls whether linking succeeds when pin mismatches occur between an instance and a reference. By default, the linking fails when there are pin mismatches. For example, when a pin exists in the instance but does not exist in the library, the linker issues an error and fails. When you set the **link_allow_design_mismatch** variable to *true*, the linker ignores the extra pin and the link succeeds. This allows you to gather useful information when part of a design is in the early stages of development.

Common causes of mismatches include:

1. A pin of an instance does not exist in the reference.
2. Bus widths of the instance and the reference are different.

When a mismatch occurs, the reference always wins. For the first case, the pin of the instance is ignored. In the second case, all the extra bits in the instance are ignored and the bus width in the linked design is made the same as the bus width of the reference.

Note that for bus-width mismatch, the LSB of the instance is mapped to the LSB of the reference, all extra bits of the instance are ignored, and all extra bits of the reference are left dangling.

To determine the current value of this variable, run one of the following commands:

printvar link_allow_design_mismatch

echo $link_allow_design_mismatch

To report the mismatches, use the **report_design_mismatch** command.

SEE ALSO

printvar(2)
report_design_mismatch(2)

link_create_black_boxes

Enables design linking to automatically convert unresolved references into black boxes.

TYPE

Boolean

DEFAULT

true

DESCRIPTION

By default, this variable is set to **true**, and design linking automatically converts each unresolved reference into a black box, an empty cell with no timing arcs. The result is a completely linked design on which you can analysis.

If you set this variable to **false**, unresolved references remain unresolved, and most analysis commands cannot work.

SEE ALSO

link_design(2)
search_path(3)

link_library

This is a synonym for the **link_path** variable.

SEE ALSO

link_path(3)

link_path

Specifies a list of libraries, design files, and library files used during linking.

TYPE

list

DEFAULT

*

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Specifies a list of libraries, design files, and library files used during linking. The **link_design** command looks at those files and tries to resolve references in the order of specified files.

The **link_path** variable can contain three different types of elements: "*", a library name, or a file name.

The "*" entry in the value of this variable indicates that **link_design** should search all the designs loaded in **ptc_shell** while trying to resolve references. Designs are searched in the order in which they were read.

For elements other than "*", **ptc_shell** searches for a library that has already been loaded. If that search fails, **ptc_shell** searches for a file name using the **search_path** variable.

The default of **link_path** is "*". To determine the current value of this variable, type **printvar link_path** or **echo $link_path**.

SEE ALSO

link_design(2)
printvar(2)
search_path(3)

link_path_per_instance

Overrides the default link path for selected leaf cell or hierarchical cell instances.

TYPE

list

DEFAULT

(empty)

DESCRIPTION

This variable, which takes effect only if set before linking the current design, overrides the default **link_path** variable for selected leaf cell or hierarchical cell instances. The format is a list of lists. Each sublist consists of a pair of elements: a set of instances, and a **link_path** specification that should be used for and within these instances. For example,

```
set link_path {* lib1.db}
set link_path_per_instance [list
  [list {ucore} {* lib2.db}]
  [list {ucore/usubblk} {* lib3.db}]]
```

Entries are used to link the specified level and below. If a given block matches multiple entries in the per-instance list, the more specific entry overrides the more general entry. In the preceding example:

1. lib3.db would be used to link blocks 'ucore/usubblk' and below.

2. lib2.db would be used to link 'ucore' and below (except within 'ucore/subblk').

3. lib1.db would be used for the remainder of the design (everything except within 'ucore').

The default value of the **link_path_per_instance** variable is an empty list, meaning that the feature is disabled.

SEE ALSO

link_design(2)
link_path(3)
link_path_per_instance(3)

net_attributes

Provides a description of the predefined attributes for nets.

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Attributes are properties assigned to objects such as pins, cells and nets. Definitions for net attributes are provided in the subsections that follow.

Attributes are informational, or "read-only." You cannot set the value of attributes designed by the application.

To determine the value of an attribute, use the **get_attribute** command. To see a list of all attributes available for a class of objects use the **list_attributes -application** command.

Net Attributes

base_name

A string attribute for the name of a net. This name does not include the instance path prefix.

full_name

A string attribute for the name of a net. This name includes a prefix for the instance path to this net.

is_global

A Boolean attribute. Possible values are "true" and "false". Global nets are those tied to logic 0 and logic 1.

is_ideal

A Boolean attribute that is set with the **set_ideal_network** command.

is_design_mismatch

A Boolean attribute that is annotated by the linker indicating that the net is connected to pins that are inconsistent with its master module.

object_class

A string attribute with the value of "net". Each object class has a different object class value. The possible object classes are: design, cell, pin, port, net, lib, lib_cell, lib_pin, clock, scenario, input_delay, output_delay, timing_exception, timing_arc, lib_timing_arc.

SEE ALSO

get_attribute(2)
list_attributes(2)

optimize_parallel_arcs

Compress multiple cell arcs with same timimg sense between a from and to pin during linking the design

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

When set to *true*, this variable allow the compression of multiple parallel timing arcs with same sense between a from and a to pin in a cell, causing improvement in PTC memory and runtime. However, the number of timing paths reported by the debug commands like analyze_paths may decrease since the tool will consider a single path exists between the two pins.

To determine the current value of this variable, enter one of the following commands:

ptc_shell> **printvar optimize_parallel_arcs**

or

ptc_shell> **echo optimize_parallel_arcs**

SEE ALSO

link_design(2)

output_delay_attributes

Provides a description of the predefined attributes for output delay objects.

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Attributes are properties assigned to objects such as pins, cells and nets. Definitions for output_delay attributes are provided in the subsections that follow.

Attributes are informational, or "read-only." You cannot set the value of attributes designed by the application.

To determine the value of an attribute, use the **get_attribute** command. To see a list of all attributes available for a class of objects use the **list_attributes -application** command.

Output_Delay Attributes

clock_name

A string attribute representing the name of the relative clock.

file_line_info

A string attribute. If it exists, the value contains one or more Tcl or SDC source file name and line number pairs of the commands which were used to create this object.

full_name

A string attribute representing a unique name for the output delay object.

is_clock_fall

A Boolean attribute. The value is "true" if this output_delay is relative to the falling edge of the clock, "false" otherwise.

is_level_sensitive

A Boolean attribute. The value is "true" if this output_delay represents a level-sensitive startpoint, "false" otherwise.

is_network_latency_included

A Boolean attribute. The value is "true" if this output_delay includes clock network latency, "false" otherwise.

is_source_latency_included

A Boolean attribute. The value is "true" if this output_delay includes clock source latency, "false" otherwise.

max_fall

A float attribute representing the maximum fall delay value.

max_rise

A float attribute representing the maximum rise delay value.

min_fall

A float attribute representing the minimum fall delay value.

min_rise

A float attribute representing the minimum rise delay value.

object_class

A string attribute with the value of "output_delay". Each object class has a different object class value.

SEE ALSO

set_output_delay(2)
get_attribute(2)
list_attributes(2)

pin_attributes

Lists the predefined attributes for pins.

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Attributes are properties assigned to objects such as pins, cells and nets. Definitions for pin attributes are provided in the subsections that follow.

Attributes are informational, or "read-only." You cannot set the value of attributes designed by the application.

To determine the value of an attribute, use the **get_attribute** command. To see a list of all attributes available for a class of objects use the **list_attributes -application** command.

Pin Attributes

case_sources

A collection of pins or ports that are the source of the case value on the given pin. The sources of a case value are the locations where the user case or netlist constants have been added. For example, a port "PE" fans out to a pin "u1/A" through only buffers and a case value of 0 is set on port "PE". The case_sources attribute for pin u1/A will be port "PE". This attribute is only defined for a pin if there is a case_value on that pin.

case_value

Specifies the value of case on this pin. The possible values are "0" or "1". This value might come from user-defined case or netlist defined constant value. The value might be because a case value was directly set on this pin or because the case value was propagated through logic to this pin. If there is no case assigned or propagated to this pin this attribute is not defined.

clock_latency_fall_max

A float attribute representing the user-specified clock network latency for max fall. This attribute is only defined on those pins where **set_clock_latency** has been directly set.

clock_latency_fall_min

A float attribute representing the user-specified clock network latency for min fall. This attribute is only defined on those pins where **set_clock_latency** has been directly set.

clock_latency_rise_max

A float attribute representing the user-specified clock network latency for max rise. This attribute is only defined on those pins where **set_clock_latency** has been directly set.

clock_latency_rise_min

A float attribute representing the user-specified clock network latency for min rise. This attribute is only defined on those pins where **set_clock_latency** has been directly set.

clock_source_latency_early_fall_max

A float attribute representing the user-specified clock network latency for early max fall. This attribute is only defined on those pins where **set_clock_latency -source** has been directly set.

clock_source_latency_early_fall_min

A float attribute representing the user-specified clock network latency for early min fall. This attribute is only defined on those pins

where **set_clock_latency -source** has been directly set.

clock_source_latency_early_rise_max

A float attribute representing the user-specified clock network latency for early max rise. This attribute is only defined on those pins where **set_clock_latency -source** has been directly set.

clock_source_latency_early_rise_min

A float attribute representing the user-specified clock network latency for early min rise. This attribute is only defined on those pins where **set_clock_latency -source** has been directly set.

clock_source_latency_late_fall_max

A float attribute representing the user-specified clock network latency for late max fall. This attribute is only defined on those pins where **set_clock_latency -source** has been directly set.

clock_source_latency_late_fall_min

A float attribute representing the user-specified clock network latency for late min fall. This attribute is only defined on those pins where **set_clock_latency -source** has been directly set.

clock_source_latency_late_rise_max

A float attribute representing the user-specified clock network latency for late max rise. This attribute is only defined on those pins where **set_clock_latency -source** has been directly set.

clock_source_latency_late_rise_min

A float attribute representing the user-specified clock network latency for late min rise. This attribute is only defined on those pins where **set_clock_latency -source** has been directly set.

clocks

A collection of clock objects that this pin is in the clock network of. The clocks in the collection are just those clocks that this pin fans out directly from the clock sources. It does not include the master clocks of any generated clock networks that this pin is in.

clocks_with_sense

A string attribute in the form of a list of clock names and clock senses this pin is in the clock network of. The possible clock senses are: "positive", "negative", "rise_triggered_high_pulse", "fall_triggered_high_pulse", "rise_triggered_low_pulse", "fall_triggered_low_pulse". For example, the following script:

```
   foreach_in_collection pin $pinList {
set pinName [get_attribute $pin full_name]
      set senses [ get_attribute $pin clocks_with_sense]
      echo "For pin: " $pinName " " $senses;
   }
```

might produce output such as:

```
For pin:  mux/A   { { "clk" "negative" } }
For pin:  mux/B   { { "clk" "positive" } }
For pin:  mux/Z   { { "clk" "negative" } { "clk" "positive" } }
```

constant_value

A string attribute for the constant value on this pin. The possible values are "0" or "1". Constant values are caused by a logic constant in the netlist or by a library cell such as a pullup or pulldown having a constant logic function. A constant value is on the initial pin and on any pins that have their case propagated as a result of that. For pins with the attribute constant_value defined, the attribute case_value will also be defined and have the same value.

direction

A string attribute for the direction of a pin. Possible values are *in*, *out*, *inout*, or *unknown*.

disable_timing

A Boolean attribute that is true if the pin is disabled when you use set_disable_timing.

full_name

A string attribute returning the full instance path name to the pin.

hold_uncertainty

A float attribute representing the user-specified clock uncertainty for hold. This attribute exists only if **set_clock_uncertainty** was specified on this object.

ideal_latency_max_fall

ideal_latency_max_rise

ideal_latency_min_fall

ideal_latency_min_rise

Floating point attributes set with the command **set_ideal_latency**.

ideal_transition_max_fall

ideal_transition_max_rise

ideal_transition_min_fall

ideal_transition_min_rise

Floating point attributes set with the command **set_ideal_transition**.

is_async_pin

A Boolean attribute that is true if this pin is a preset or clear pin.

is_clear_pin

A Boolean attribute that is true if this pin is a clear (reset) pin.

is_clock_gating_clock

A Boolean attribute that is true if a pin is a clock pin of a clock-gating cell.

is_clock_gating_enable

A Boolean attribute that is true if a pin is an enable pin of a clock-gating cell.

is_clock_gating_pin

A Boolean attribute that is true if a pin is a pin of a clock-gating cell.

is_clock_pin

A Boolean attribute that is true for clock pins of registers. A clock pin is any pin that has a timing check from it to a data pin.

is_clock_used_as_clock

A Boolean attribute that is true if this pin is part of at least one clock network and the clock network fanout of this pin reaches at least one register clock pin or output port.

is_clock_used_as_data

A Boolean attribute that is true if this pin is part of at least one clock network and the clock network fanout of this pin reaches at least one register data pin. If the attribute values for is_clock_used_as_data is TRUE and is_clock_used_as_clock is FALSE, the clocks through this pin only acts a data and never as clock.

is_data_pin

A Boolean attribute that is true for data pins of registers. A data pin is any pin that has a timing check to it. Data pins are legal endpoint for timing paths.

is_fall_edge_triggered_clock_pin

A Boolean attribute that is true for clock pins of registers with falling edge-triggered behavior.

is_fall_edge_triggered_data_pin

A Boolean attribute that is true for data pins of registers with falling edge-triggered behavior.

is_hierarchical

A Boolean attribute that is true if the pin's cell is hierarchical.

is_ideal

A Boolean attribute that is set with the command **set_ideal_network**.

is_design_mismatch

A Boolean attribute that is annotated by the linker indicating this pin is inconsistent with its master module.

is_mux_select_pin

A Boolean attribute that is true if the pin a select pin of a mux cell.

is_negative_level_sensitive_clock_pin

A Boolean attribute that is true for clock pins of latches with negative level-sensitive behavior.

is_negative_level_sensitive_data_pin

A Boolean attribute that is true for data pins of latches with negative level-sensitive behavior.

is_positive_level_sensitive_clock_pin

A Boolean attribute that is true for clock pins of latches with positive level-sensitive behavior.

is_positive_level_sensitive_data_pin

A Boolean attribute that is true for data pins of latches with positive level-sensitive behavior.

is_preset_pin

A Boolean attribute that is true if this pin is an asynchronous preset pin.

is_rise_edge_triggered_clock_pin

A Boolean attribute that is true for clock pins of registers with rising edge-triggered behavior.

is_rise_edge_triggered_data_pin

A Boolean attribute that is true for data pins of registers with rising edge-triggered behavior.

is_three_state

A Boolean attribute that is true if this pin is a three-state output (can become high impedance when not enabled).

is_three_state_enable_pin

A Boolean attribute. Possible values are "true" and "false". A pin is a three-state enable pin if there arethree-state enable and or three-state disable arcs from that pin.

is_three_state_output_pin

A Boolean attribute that is true if this pin is a three-state output (can become high impedance when not enabled).

lib_pin_name

A string attribute for the name of the library pin.

max_capacitance

A floating point attribute set with **set_max_capacitance**.

max_time_borrow

A floating point attribute set with **set_max_time_borrow**.

max_transition

A floating point attribute set with **set_max_transition**.

object_class

A string attribute with the value of "pin". Each object class has a different object class value. The possible object classes are: design, cell, pin, port, net, lib, lib_cell, lib_pin, clock, scenario, input_delay, output_delay, timing_exception, timing_arc, lib_timing_arc.

pin_direction

A string attribute for the direction of a pin. Possible values are *in*, *out*, *inout*, or *unknown*. This is the same as "direction". This variable is supported for compatibility with other tools. The "direction" attribute is supported for both ports and pins and therefore easier to use for heterogeneous collections of pins and ports.

potential_clocks

A collection of clock objects that this pin would be in the clock network of, except these clocks are blocked by some constraint. The command analyze_clock_networks can be used to find the reason these clocks are blocked.

setup_uncertainty

A float attribute representing the user-specified clock uncertainty for setup. This attribute exists only if **set_clock_uncertainty** was specified on this object.

temperature_max

A float attribute that specifies the maximum temperature value, in degrees Celsius, for the pin.

temperature_min

A float attribute that specifies the min temperature value, in degrees Celsius, for the pin.

user_case_value

A string attribute for the case value set by the command **set_case_analysis**. This attribute is only defined on those pins that set_case_analysis has been directly set on. Use the attribute "case_value" for both directly set case values and propagated case values.

user_clock_sense

A string attribute for the sense set by the command **set_clock_sense**. This attribute is only define on those pins where the **set_clock_sense** command was used directly.

SEE ALSO

find(2)
get_attribute(2)
list_attributes(2)
set_clock_uncertainty(2)

port_attributes

Lists the predefined attributes for ports.

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Attributes are properties assigned to objects such as ports, cells and nets. Definitions for port attributes are provided in the subsections that follow.

Attributes are informational, or "read-only." You cannot set the value of attributes designed by the application.

To determine the value of an attribute, use the **get_attribute** command. To see a list of all attributes available for a class of objects use the **list_attributes -application** command.

Port Attributes

case_sources

A collection of pins or ports that are the source of the case value on the given port. The sources of a case value are the locations where the user case or netlist constants have been added. For example, a port "PE" fans out to a port "OUT34" through only buffers and a case value of 0 is set on port "PE". The case_sources attribute for port "OUT34 will be port "PE". This attribute is only defined for a port if there is a case_value on that port.

case_value

Specifies the value of case on this port. The possible values are "0" or "1". This value might come from user-defined case or netlist defined constant value. The value might be because a case value was directly set on this port or because the case value was propagated through logic to this port. If there is no case assigned or propagated to this port this attribute is not defined.

clock_latency_fall_max

A float attribute representing the user-specified clock network latency for max fall. This attribute is only defined on those ports where **set_clock_latency** has been directly set.

clock_latency_fall_min

A float attribute representing the user-specified clock network latency for min fall. This attribute is only defined on those ports where **set_clock_latency** has been directly set.

clock_latency_rise_max

A float attribute representing the user-specified clock network latency for max rise. This attribute is only defined on those ports where **set_clock_latency** has been directly set.

clock_latency_rise_min

A float attribute representing the user-specified clock network latency for min rise. This attribute is only defined on those ports where **set_clock_latency** has been directly set.

clock_source_latency_early_fall_max

A float attribute representing the user-specified clock network latency for early max fall. This attribute is only defined on those ports where **set_clock_latency -source** has been directly set.

clock_source_latency_early_fall_min

A float attribute representing the user-specified clock network latency for early min fall. This attribute is only defined on those ports

where **set_clock_latency -source** has been directly set.

clock_source_latency_early_rise_max

A float attribute representing the user-specified clock network latency for early max rise. This attribute is only defined on those ports where **set_clock_latency -source** has been directly set.

clock_source_latency_early_rise_min

A float attribute representing the user-specified clock network latency for early min rise. This attribute is only defined on those ports where **set_clock_latency -source** has been directly set.

clock_source_latency_late_fall_max

A float attribute representing the user-specified clock network latency for late max fall. This attribute is only defined on those ports where **set_clock_latency -source** has been directly set.

clock_source_latency_late_fall_min

A float attribute representing the user-specified clock network latency for late min fall. This attribute is only defined on those ports where **set_clock_latency -source** has been directly set.

clock_source_latency_late_rise_max

A float attribute representing the user-specified clock network latency for late max rise. This attribute is only defined on those ports where **set_clock_latency -source** has been directly set.

clock_source_latency_late_rise_min

A float attribute representing the user-specified clock network latency for late min rise. This attribute is only defined on those ports where **set_clock_latency -source** has been directly set.

clocks

A collection of clock objects that this port is in the clock network of. The clocks in the collection are just those clocks that this port fans out directly from the clock sources. It does not include the master clocks of any generated clock networks that this port is in.

clocks_with_sense

A string attribute in the form of a list of clock names and clock senses that propagate to this port. The possible clock senses are: "positive", "negative", "rise_triggered_high_pulse", "fall_triggered_high_pulse", "rise_triggered_low_pulse", "fall_triggered_low_pulse".

constant_value

A string attribute for the constant value on this port. The possible values are "0" or "1". Constant values are caused by a logic constant in the netlist or by a library cell such as a pullup or pulldown having a constant logic function. A constant value is on the initial port and on any ports that have their case propagated as a result of that constant value. For ports with the attribute constant_value defined, the attribute case_value will also be defined and have the same value.

direction

A string attribute for the direction of a port. Possible values are *in*, *out*, *inout*, or *unknown*.

disable_timing

A Boolean attribute that is true if the port is disabled when you use set_disable_timing.

drive_resistance_fall_max

drive_resistance_fall_min

drive_resistance_rise_max

drive_resistance_rise_min

Floating point attributes for retrieving the values set with the **set_drive** command.

driving_cell_rise_max

driving_cell_rise_min

driving_cell_fall_max

driving_cell_fall_min

String attributes for retrieving the cell names set with the **set_driving_cell** command.

driving_cell_pin_rise_max

driving_cell_pin_rise_min

driving_cell_pin_fall_max

driving_cell_pin_fall_min

String attributes for retrieving the pin names set with the **set_driving_cell -pin** command.

driving_cell_from_pin_rise_max

driving_cell_from_pin_rise_min

driving_cell_from_pin_fall_max

driving_cell_from_pin_fall_max

String attributes for retrieving the from pin names set with the **set_driving_cell -from_pin** command.

driving_cell_library_rise_max

driving_cell_library_rise_min

driving_cell_library_fall_max

driving_cell_library_fall_min

String attributes for retrieving the library names set with the **set_driving_cell -library** command.

driving_cell_max_rise_itrans_rise

driving_cell_max_rise_itrans_fall

driving_cell_max_fall_itrans_rise

driving_cell_max_fall_itrans_fall

driving_cell_min_rise_itrans_rise

driving_cell_min_rise_itrans_fall

driving_cell_min_fall_itrans_rise

driving_cell_min_fall_itrans_fall

Floating attributes for retrieving the input_transition values set with the **set_driving_cell** command.

driving_cell_multiply_by

A floating attribute for retrieving the value set with the **set_driving_cell -multiply_by** command.

driving_cell_dont_scale

A Boolean attribute that is set with **set_driving_cell** command.

driving_cell_no_design_rule

A Boolean attribute that is set with **set_driving_cell** command.

fanout_load

A floating point attribute set with **set_fanout_load**.

full_name

A string attribute for the name of a port.

hold_uncertainty

A float attribute.

ideal_latency_max_fall

ideal_latency_max_rise

ideal_latency_min_fall

ideal_latency_min_rise

Floating point attributes set with the command **set_ideal_latency**.

ideal_transition_max_fall

ideal_transition_max_rise

ideal_transition_min_fall

ideal_transition_min_rise

Floating point attributes set with the command **set_ideal_transition**.

input_transition_fall_max

input_transition_fall_min

input_transition_rise_max

input_transition_rise_min

Floating point attributes set with the command **set_input_transition**.

is_clock_used_as_clock

A Boolean attribute that is true if this port is part of at least one clock network and the clock network fanout of this port reaches at least one register clock pin or output port.

is_clock_used_as_data

A Boolean attribute that is true if this port is part of at least one clock network and the clock network fanout of this port reaches at least one register data pin. If the attribute values for is_clock_used_as_data is TRUE and is_clock_used_as_clock is FALSE, the clocks from this port only acts a data and never as clock.

is_design_mismatch

A Boolean attribute that is annotated by linker indicating this port is involved in connections that are not consistent.

is_ideal

A Boolean attribute that is set with the command **set_ideal_network**.

max_capacitance

A floating point attribute set with **set_max_capacitance**.

max_fanout

A floating point attribute set with **set_max_fanout**.

max_transition

A floating point attribute set with **set_max_transition**.

object_class

A string attribute with the value of "port". Each object class has a different object class value. The possible object classes are: design, cell, pin, port, net, lib, lib_cell, lib_pin, clock, scenario, input_delay, output_delay, timing_exception, timing_arc,

lib_timing_arc.

pin_capacitance_max_fall

pin_capacitance_max_rise

pin_capacitance_min_fall

pin_capacitance_min_rise

Floating point attributes set with the command **set_load**.

port_direction

A string attribute for the direction of a port. Possible values are *in*, *out*, *inout*, or *unknown*. This is the same as "direction". This variable is supported for compatibility with other tools. The "direction" attribute is supported for both ports and pins and therefore easier to use for heterogeneous collections of pins and ports.

potential_clocks

A collection attribute. Contents of the collection are clocks that did not propagate to this port, but potentially would have propagated if they were not blocked by other constraints. The command analyze_clock_networks can be used to find the reason these clocks are blocked.

setup_uncertainty

A float attribute.

temperature_max

A float attribute that specifies the maximum temperature value, in degrees Celsius, for the port.

temperature_min

A float attribute that specifies the minimum temperature value, in degrees Celsius, for the port.

user_case_value

A string attribute for the case value set by the command **set_case_analysis**. This attribute is only defined on those ports where set_case_analysis has been directly set. Use the attribute "case_value" for both directly set case values and propagated case values.

user_clock_sense

A string attribute for the sense set by the command **set_clock_sense**. This attribute is only define on those ports where the **set_clock_sense** command was used directly.

wire_capacitance_max_fall

wire_capacitance_max_rise

wire_capacitance_min_fall

wire_capacitance_min_rise

Floating point attributes set with the command **set_load**.

SEE ALSO

find(2)
get_attribute(2)
list_attributes(2)

port_search_in_current_instance

Controls whether the **get_ports command** can get ports of current instance.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

When set to *true*, the **get_ports** command gets ports of the current instance; when set to *false* (the default), it gets the port of the current design.

To determine the current value of this variable, type one of the following: **printvar port_search_in_current_instance echo $port_search_in_current_instance**

SEE ALSO

get_ports(2)

pt_synopsys_root

Specifies installation root containing constraint consistency.

TYPE

string

DEFAULT

none

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

This variable can be set to give a location of the PrimeTime installation root. This location is used to check that the version of PrimeTime supports the constraint commands given to constraint consistency.

To determine the current value of this variable, use **printvar pt_synopsys_root**.

For example:

```
set_app_var pt_synopsys_root /u/release/primetime/M-2016.12
```

The **analyze_design** command can perform rule checks for whether commands are compatible across the multiple products. These are rules CMP_0001 for commands and CMP_0002 for options of commands.

SEE ALSO

icc_synopsys_root(3)
dc_synopsys_root(3)
analyze_design(2)

query_objects_format

TYPE

string

DEFAULT

application specific

DESCRIPTION

This variable sets the format that the **query_objects** command uses to print its result. There are two supported formats: Legacy and Tcl.

The Legacy format looks like this:

```
{"or1", "or2", "or3"}
```

The Tcl format looks like this:

```
{or1 or2 or3}
```

Please see the man page for **query_objects** for complete details.

SEE ALSO

query_objects(2)

report_default_significant_digits

Sets the default number of significant digits used to display values in reports.

TYPE

int

DEFAULT

2

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

The **report_default_significant_digits** variable sets the default number of significant digits for many reports. Allowed values are 0-13; the default is 2. Some report commands (for example, the **report_timing** command) have a **-significant_digits** option that overrides the value of this variable.

Not all reports respond to this variable. Check the man page of individual reports to determine whether they support this feature.

To determine the current value of this variable, type one of the following:

printvar report_default_significant_digits

echo report_default_significant_digits

SEE ALSO

analyze_design(2)

rule_attributes

Provides a description of the predefined attributes for rules.

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Attributes are properties assigned to objects such as pins, cells, and nets. Definitions for rule attributes are provided in the subsections that follow.

Attributes are informational, or "read-only." You cannot set the value of attributes designed by the application.

To determine the value of an attribute, use the **get_attribute** command. To see a list of all attributes available for a class of objects use the **list_attributes -application** command.

Rule Attributes

description

A string attribute representing the description of a rule.

file_line_info

A string attribute. If it exists, the value contains one or more Tcl or SDC source file name and line number pairs of the commands which were used to create this object.

full_name

A string attribute for the name of a rule.

is_enabled

A Boolean attribute that is true if the rule is currently enabled.

message

A string attribute representing the message text as a list of message parts.

object_class

A string attribute with the value of "rule". Each object class has a different object class value.

parameters

A string attribute representing the parameter names as a list of strings.

properties

A string attribute representing the property name associated with a rule. Only some rules have properties associated with them. For example, rule EXD_0009 has a property "max_percent".

severity

A string attribute representing the severity of a violation of this rule. Valid values are: "info", "warning", "error", and "fatal".

violation_details

A collection of strings representing the names of the violation details that can be queried to get the details of the rule violation via get_violation_info(2).

SEE ALSO

create_rule(2)
get_attribute(2)
list_attributes(2)

rule_violation_attributes

Provides a description of the predefined attributes for rule violations.

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Attributes are properties assigned to objects such as pins, cells and nets. Definitions for rule attributes are provided in the subsections that follow.

Attributes are informational, or "read-only." You cannot set the value of attributes designed by the application.

To determine the value of an attribute, use the **get_attribute** command. To see a list of all attributes available for a class of objects use the **list_attributes -application** command.

Rule Attributes

design

A string attribute representing the design where the violation is generated.

is_waived

A Boolean attribute representing if the violation is waived.

message

A string attribute representing the rule violation message.

object_class

A string attribute with the value of "rule_violation".

rule

A string attribute representing the name of the rule.

scenario

A string attribute representing the scenario in which the violation is generated.

second_design

A string attribute representing the secondary design where the violation is generated. Note that these attributes are available for inter-scenario, or inter-design rules.

second_scenario

A string attribute representing the secondary scenario where the violation is generated. Note that these attributes are available for inter-scenario, or inter-design rules.

SEE ALSO

report_attribute(2)
get_attribute(2)

list_attributes(2)

ruleset_attributes

Provides a description of the predefined attributes for rulesets.

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Attributes are properties assigned to objects such as pins, cells and nets. Definitions for ruleset attributes are provided in the subsections that follow.

Attributes are informational, or "read-only." You cannot set the value of attributes designed by the application.

To determine the value of an attribute, use the **get_attribute** command. To see a list of all attributes available for a class of objects use the **list_attributes -application** command.

Ruleset Attributes

full_name

A string attribute for the name of a ruleset.

object_class

A string attribute with the value of "ruleset". Each object class has a different object class value.

SEE ALSO

create_ruleset(2)
get_attribute(2)
list_attributes(2)

scenario_attributes

Provides a description of the predefined attributes for scenarios.

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Attributes are properties assigned to objects such as pins, cells and nets. Definitions for scenario attributes are provided in the subsections that follow.

Attributes are informational, or "read-only." You cannot set the value of attributes designed by the application.

To determine the value of an attribute, use the **get_attribute** command. To see a list of all attributes available for a class of objects use the **list_attributes -application** command.

Scenario Attributes

file_line_info

A string attribute. If it exists, the value contains one or more Tcl or SDC source file name and line number pairs of the commands which were used to create this object.

full_name

A string attribute for the name of a scenario.

is_current

A Boolean attribute. The value is "true" if this scenario is the current scenario, "false" otherwise.

is_default

A Boolean attribute. The value is "true" if this scenario is the default scenario, "false" otherwise.

object_class

A string attribute with the value of "scenario". Each object class has a different object class value.

SEE ALSO

create_scenario(2)
get_attribute(2)
list_attributes(2)

sdc_version

Specifies the SDC version that was written. Use in context of reading an SDC file.

TYPE

string

DEFAULT

latest version

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

The **sdc_version** variable is meaningful only within the context of reading an SDC file. Setting it outside an SDC file has no effect, other than to produce an informational message.

The **write_sdc** command writes a command to the SDC file to set the **sdc_version** variable to the version that was written. There is no user-control over the version of SDC that is written. The most current version is written. When **read_sdc** reads the SDC file, it validates the version specified in the file (if present) with the version requested by the command.

SEE ALSO

read_sdc(2)
write_sdc(2)

search_path

Shows a list of directory names that contain design and library files that are specified without directory names.

TYPE

list

DEFAULT

"" (empty)

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

A list of directory names that specifies directories to be searched for design and library files that are specified without directory names. Normally, **search_path** is set to a central library directory. The default of **search_path** is the empty string, "". The **read_db** and **link_design** commands particularly depend on **search_path**.

You can cause the **source** command to search for scripts using **search_path**, by setting the **sh_source_uses_search_path** variable to *true*.

To determine the current value of this variable, type **printvar search_path** or **echo $search_path**.

SEE ALSO

link_design(2)
printvar(2)
read_db(2)
sh_source_uses_search_path(3)
source(2)

selection_logging_no_core_action

This variable determines how the *change_selection_no_core* command behaves.

TYPE

String

DEFAULT

""

DESCRIPTION

This variable determines how the Tcl command *change_selection_no_core* behaves.

SEE ALSO

change_selection_no_core(2)

selection_logging_too_many_objects_action

This variable determines how the *change_selection_too_many_objects* command behaves.

TYPE

String

DEFAULT

""

DESCRIPTION

This variable determines how the *change_selection_too_many_objects* Tcl command behaves.

SEE ALSO

change_selection_too_many_objects(2)

sh_allow_tcl_with_set_app_var

Allows the **set_app_var** and **get_app_var** commands to work with application variables.

TYPE

string

DEFAULT

application specific

DESCRIPTION

Normally the **get_app_var** and **set_app_var** commands only work for variables that have been registered as application variables. Setting this variable to **true** allows these commands to set a Tcl global variable instead.

These commands issue a CMD-104 error message for the Tcl global variable, unless the variable name is included in the list specified by the **sh_allow_tcl_with_set_app_var_no_message_list** variable.

SEE ALSO

get_app_var(2)
set_app_var(2)
sh_allow_tcl_with_set_app_var_no_message_list(2)

sh_allow_tcl_with_set_app_var_no_message_list

Suppresses CMD-104 messages for variables in this list.

TYPE

string

DEFAULT

application specific

DESCRIPTION

This variable is consulted before printing the CMD-104 error message, if the **sh_allow_tcl_with_set_app_var** variable is set to **true**. All variables in this Tcl list receive no message.

SEE ALSO

get_app_var(2)
set_app_var(2)
sh_allow_tcl_with_set_app_var(2)

sh_arch

Indicates the system architecture of your machine.

TYPE

string

DEFAULT

platform-dependent

DESCRIPTION

The **sh_arch** variable is set by the application to indicate the system architecture of your machine. Examples of machines being used are sparcOS5, amd64, and so on. This variable is read-only.

sh_auto_log_generation

Specifies whether the application should automatically generate session log files.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

If the variable is set to *true* in the .synopsys_gca.setup file and the **sh_output_log_file** variable is not set, application chooses a file name based on current time stamp in the current working directory and automatically sets the variable. Therefore, the session log is created in that file.

This variable can be set only in a setup file.

To determine the current value of this variable, type **printvar sh_auto_log_generation**.

SEE ALSO

printvar(2)
sh_output_log_file(3)

sh_command_abbrev_mode

Sets the command abbreviation mode for interactive convenience.

TYPE

string

DEFAULT

application specific

DESCRIPTION

This variable sets the command abbreviation mode as an interactive convenience. Script files should not use any command or option abbreviation, because these files are then susceptible to command changes in subsequent versions of the application.

Although the default value is **Anywhere**, it is recommended that the site startup file for the application set this variable to **Command-Line-Only**. It is also possible to set the value to **None**, which disables abbreviations altogether.

To determine the current value of this variable, use the **get_app_var sh_command_abbrev_mode** command.

SEE ALSO

sh_command_abbrev_options(3)
get_app_var(2)
set_app_var(2)

sh_command_abbrev_options

Turns off abbreviation of command dash option names when false.

TYPE

boolean

DEFAULT

application specific

DESCRIPTION

When command abbreviation is currently off (see sh_command_abbrev_mode) then setting this variable to false will also not allow abbreviation of command dash options. This variable also impacts abbreviation of the values specified to command options that expect values to be one of an allowed list of values.

This variable exists to be backward compatible with previous tool releases which always allowed abbreviation of command dash options and option values regardless of the command abbreviation mode.

It is recommended to set the value of this variable to false.

To determine the current value of this variable, use the **get_app_var sh_command_abbrev_options** command.

SEE ALSO

sh_command_abbrev_mode(3)
get_app_var(2)
set_app_var(2)

sh_command_log_file

Specifies the name of the file to which the application logs the commands you executed during the session.

TYPE

string

DEFAULT

empty string

DESCRIPTION

This variable specifies the name of the file to which the application logs the commands you run during the session. By default, the variable is set to an empty string, indicating that the application's default command log file name is to be be used. If a file named by the default command log file name cannot be opened (for example, if it has been set to read only access), then no logging occurs during the session.

This variable can be set at any time. If the value for the log file name is invalid, the variable is not set, and the current log file persists.

To determine the current value of this variable, use the **get_app_var sh_command_log_file** command.

SEE ALSO

get_app_var(2)
set_app_var(2)

sh_continue_on_error

Allows processing to continue when errors occur during script execution with the **source** command.

TYPE

Boolean

DEFAULT

application specific

DESCRIPTION

It is recommended to use the **-continue_on_error** option to the **source** command instead of this variable because that option only applies to a single script, and not the entire application session.

When set to **true**, the **sh_continue_on_error** variable allows processing to continue when errors occur. Under normal circumstances, when executing a script with the **source** command, Tcl errors (syntax and semantic) cause the execution of the script to terminate.

When **sh_continue_on_error** is set to **false**, script execution can also terminate due to new error and warning messages based on the value of the **sh_script_stop_severity** variable.

To determine the current value of the **sh_continue_on_error** variable, use the **get_app_var sh_continue_on_error** command.

SEE ALSO

get_app_var(2)
set_app_var(2)
source(2)
sh_script_stop_severity(3)

sh_deprecated_is_error

Raise a Tcl error when a deprecated command is executed.

TYPE

Boolean

DEFAULT

application specific

DESCRIPTION

When set this variable causes a Tcl error to be raised when an deprecated command is executed. Normally only a warning message is issued.

SEE ALSO

get_app_var(2)
set_app_var(2)

sh_dev_null

Indicates the current null device.

TYPE

string

DEFAULT

platform dependent

DESCRIPTION

This variable is set by the application to indicate the current null device. For example, on UNIX machines, the variable is set to /**dev**/**null**. This variable is read-only.

SEE ALSO

get_app_var(2)

sh_enable_line_editing

Enables the command line editing capabilities in constraint consistency.

TYPE

Boolean

DEFAULT

true

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

If set to true it enables advanced unix like shell capabilities.

This variable needs to be set in .synopsys_pt.setup file to take effect.

Key Bindings

The **list_key_bindings** command displays current key bindings and the edit mode. To change the edit mode, variable **sh_line_editing_mode** can be set in either the .synopsys_pt.setup file or directly in the shell.

Command Completion

The editor can complete commands, options, variables and files given a unique abbreviation. You need to type part of a word and press the Tab key to get the complete command, variable, or file. For command options, enter '-' and press Tab key to get the options list.

If no match is found, the terminal bell rings. If the word is already complete a space is added to the end speed typing and provide a visual indicator of successful completion. Completed text pushes the rest of the line to the right. If there are multiple matches all the matching commands, options, files, or variables are listed.

Completion works in following context sensitive way :-

The first token of a command line : completes commands

Token that begins with "-" after a command : completes command arguments

After a ">", "|" or a "sh" command : completes file names

After a set, unset or printvar command : completes the variables

After '$' symbol : completes the variables

After the help command : completes command

After the man command : completes commands or variables

Any token which is not the first token and does not match any of the preceding rules : completes file names

SEE ALSO

sh_line_editing_mode(3)
list_key_bindings(2)

sh_enable_page_mode

Displays long reports one page at a time (similar to the UNIX **more** command.

TYPE

Boolean

DEFAULT

application specific

DESCRIPTION

This variable displays long reports one page at a time (similar to the UNIX **more** command), when set to **true**. Consult the man pages for the commands that generate reports to see if they are affected by this variable.

To determine the current value of this variable, use the **get_app_var sh_enable_page_mode** command.

SEE ALSO

get_app_var(2)
set_app_var(2)

sh_enable_stdout_redirect

Allows the redirect command to capture output to the Tcl stdout channel.

TYPE

Boolean

DEFAULT

application specific

DESCRIPTION

When set to **true**, this variable allows the redirect command to capture output sent to the Tcl stdout channel. By default, the Tcl **puts** command sends its output to the stdout channel.

SEE ALSO

get_app_var(2)
set_app_var(2)

sh_help_shows_group_overview

Changes the behavior of the "help" command.

TYPE

string

DEFAULT

application specific

DESCRIPTION

This variable changes the behavior of the **help** command when no arguments are specified to help. Normally when no arguments are specified an informational message with a list of available command groups is displayed.

When this variable is set to false the command groups and the commands in each group is printed instead. This variable exists for backward compatibility.

SEE ALSO

help(2)
set_app_var(2)

sh_line_editing_mode

Enables vi or emacs editing mode in the constraint consistency shell.

TYPE

string

DEFAULT

emacs

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

This variable can be used to set the command line editor mode to either vi or emacs. Valid values are emacs or vi.

Use **list_key_bindings** command to display the current key bindings and edit mode.

This variable can be set in the either the .synopsys_pt.setup file or directly in the shell. The **sh_enable_line_editing** variable must be set to *true*.

SEE ALSO

sh_enable_line_editing(3)
list_key_bindings(2)

sh_new_variable_message

Controls a debugging feature for tracing the creation of new variables.

TYPE

Boolean

DEFAULT

application specific

DESCRIPTION

The **sh_new_variable_message** variable controls a debugging feature for tracing the creation of new variables. Its primary debugging purpose is to catch the misspelling of an application-owned global variable. When set to **true**, an informational message (CMD-041) is displayed when a variable is defined for the first time at the command line. When set to **false**, no message is displayed.

Note that this debugging feature is superseded by the new **set_app_var** command. This command allows setting only application-owned variables. See the **set_app_var command man page for details.**

Other variables, in combination with **sh_new_variable_message**, enable tracing of new variables in scripts and Tcl procedures.

Warning: This feature has a significant negative impact on CPU performance when used with scripts and Tcl procedures. This feature should be used only when developing scripts or in interactive use. When you turn on the feature for scripts or Tcl procedures, the application issues a message (CMD-042) to warn you about the use of this feature.

To determine the current value of this variable, use the **get_app_var sh_new_variable_message** command.

SEE ALSO

get_app_var(2)
set_app_var(2)
sh_new_variable_message_in_proc(3)
sh_new_variable_message_in_script(3)

sh_new_variable_message_in_proc

Controls a debugging feature for tracing the creation of new variables in a Tcl procedure.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

The **sh_new_variable_message_in_proc** variable controls a debugging feature for tracing the creation of new variables in a Tcl procedure. Its primary debugging purpose is to catch the misspelling of an application-owned global variable.

Note that this debugging feature is superseded by the new **set_app_var** command. This command allows setting only application-owned variables. Please see the **set_app_var** command man page for details.

Note that the **sh_new_variable_message** variable must be set to **true** for this variable to have any effect. Both variables must be set to **true** for the feature to be enabled. Enabling the feature simply enables the **print_proc_new_vars** command. In order to trace the creation of variables in a procedure, this command must be inserted into the procedure, typically as the last statement. When all of these steps have been taken, an informational message (CMD-041) is generated for new variables defined within the procedure, up to the point that the **print_proc_new_vars** commands is executed.

Warning: This feature has a significant negative impact on CPU performance. This should be used only when developing scripts or in interactive use. When you turn on the feature, the application issues a message (CMD-042) to warn you about the use of this feature.

To determine the current value of this variable, use the **get_app_var sh_new_variable_message_in_proc** command.

SEE ALSO

get_app_var(2)
print_proc_new_vars(2)
set_app_var(2)
sh_new_variable_message(3)
sh_new_variable_message_in_script(3)

sh_new_variable_message_in_script

Controls a debugging feature for tracing the creation of new variables within a sourced script.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

The **sh_new_variable_message_in_script** variable controls a debugging feature for tracing the creation of new variables within a sourced script. Its primary debugging purpose is to catch the misspelling of an application-owned global variable.

Note that this debugging feature is superseded by the new **set_app_var** command. This command allows setting only application-owned variables. See the **set_app_var** command man page for details.

Note that the **sh_new_variable_message** variable must be set to **true** for this variable to have any effect. Both variables must be set to **true** for the feature to be enabled. In that case, an informational message (CMD-041) is displayed when a variable is defined for the first time. When **sh_new_variable_message_in_script** is set to **false** (the default), no message is displayed at the time that the variable is created. When the **source** command completes, however, you see messages for any new variables that were created in the script. This is because the state of the variables is sampled before and after the **source** command. It is not because of inter-command sampling within the script. So, this is actually a more efficient method to see if new variables were created in the script.

For example, given the following script a.tcl:

```
echo "Entering script"
set a 23
echo a = $a
set b 24
echo b = $b
echo "Exiting script"
```

When **sh_new_variable_message_in_script** is **false** (the default), you see the following when you source the script:

```
prompt> source a.tcl
Entering script
a = 23
b = 24
Exiting script
Information: Defining new variable 'a'. (CMD-041)
Information: Defining new variable 'b'. (CMD-041)
prompt>
```

Alternatively, when **sh_new_variable_message_in_script** is **true**, at much greater cost, you see the following when you source the script:

```
prompt> set sh_new_variable_message_in_script true
Warning: Enabled new variable message tracing -
```

```
        Tcl scripting optimization disabled. (CMD-042)
true
prompt> source a.tcl
Entering script
Information: Defining new variable 'a'. (CMD-041)
a = 23
Information: Defining new variable 'b'. (CMD-041)
b = 24
Exiting script
prompt>
```

Warning: This feature has a significant negative impact on CPU performance. This should be used only when developing scripts or in interactive use. When you turn on the feature, the application issues a message (CMD-042) to warn you about the use of this feature.

To determine the current value of this variable, use the **get_app_var sh_new_variable_message_in_script** command.

SEE ALSO

get_app_var(2)
set_app_var(2)
sh_new_variable_message(3)
sh_new_variable_message_in_proc(3)

sh_obsolete_is_error

Raise a Tcl error when an obsolete command is executed.

TYPE

Boolean

DEFAULT

application specific

DESCRIPTION

When set this variable causes a Tcl error to be raised when an obsolete command is executed. Normally only a warning message is issued.

Obsolete commands have no effect.

SEE ALSO

get_app_var(2)
set_app_var(2)

sh_output_log_file

Specifies the name of the file to which all application output is logged.

TYPE

string

DEFAULT

The empty string

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Specifies the name of the file to which the application logs all output during the session.

If the variable is not set in the .synopsys_gca.setup file but the **sh_auto_log_generation** variable is set to *true*, the application chooses a file name based on the current time stamp in the current working directory and automatically sets the variable. Therefore, the session log is created in that file.

This variable can be set only in a setup file.

To determine the current value of this variable, type **printvar sh_output_log_file**.

SEE ALSO

printvar(2)
sh_auto_log_generation(3)

sh_product_version

Indicates the version of the application currently running.

TYPE

string

DESCRIPTION

This variable is set to the version of the application currently running. The variable is read only.

To determine the current value of this variable, use the **get_app_var sh_product_version** command.

SEE ALSO

get_app_var(2)

sh_script_stop_severity

Indicates the error message severity level that would cause a script to stop running before it completes.

TYPE

string

DEFAULT

application specific

DESCRIPTION

When a script is run with the **source** command, there are several ways to get it to stop running before it completes. One is to use the **sh_script_stop_severity** variable. This variable can be set to **none**, **W**, or **E**.

- When set to **E**, the generation of one or more error messages by a command causes a script to stop.
- When set to **W**, the generation of one or more warning or error messages causes a script to stop.
- When set to **none**, the generation messages does not cause the script to stop.

Note that **sh_script_stop_severity** is ignored if **sh_continue_on_error** is set to **true**.

To determine the current value of this variable, use the **get_app_var sh_script_stop_severity** command.

SEE ALSO

get_app_var(2)
set_app_var(2)
source(2)
sh_continue_on_error(3)

sh_source_emits_line_numbers

Indicates the error message severity level that causes an informational message to be issued, listing the script name and line number where that message occurred.

TYPE

string

DEFAULT

application specific

DESCRIPTION

When a script is executed with the **source** command, error and warning messages can be emitted from any command within the script. Using the **sh_source_emits_line_numbers** variable, you can help isolate where errors and warnings are occurring.

This variable can be set to **none**, **W**, or **E**.

- When set to **E**, the generation of one or more error messages by a command causes a CMD-082 informational message to be issued when the command completes, giving the name of the script and the line number of the command.
- When set to **W**, the generation of one or more warning or error messages causes a the CMD-082 message.

The setting of **sh_script_stop_severity** affects the output of the CMD-082 message. If the setting of **sh_script_stop_severity** causes a CMD-081 message, then it takes precedence over CMD-082.

To determine the current value of this variable, use the **get_app_var sh_source_emits_line_numbers** command.

SEE ALSO

get_app_var(2)
set_app_var(2)
source(2)
sh_continue_on_error(3)
sh_script_stop_severity(3)
CMD-081(n)
CMD-082(n)

sh_source_logging

Indicates if individual commands from a sourced script should be logged to the command log file.

TYPE

Boolean

DEFAULT

application specific

DESCRIPTION

When you source a script, the **source** command is echoed to the command log file. By default, each command in the script is logged to the command log file as a comment. You can disable this logging by setting **sh_source_logging** to **false**.

To determine the current value of this variable, use the **get_app_var sh_source_logging** command.

SEE ALSO

get_app_var(2)
set_app_var(2)
source(2)

sh_source_uses_search_path

Indicates if the **source** command uses the **search_path** variable to search for files.

TYPE

Boolean

DEFAULT

application specific

DESCRIPTION

When this variable is set to \ftrue the **source** command uses the **search_path** variable to search for files. When set to **false**, the **source** command considers its file argument literally.

To determine the current value of this variable, use the **get_app_var sh_source_uses_search_path** command.

SEE ALSO

get_app_var(2)
set_app_var(2)
source(2)
search_path(3)

sh_tcllib_app_dirname

Indicates the name of a directory where application-specific Tcl files are found.

TYPE

string

DESCRIPTION

The **sh_tcllib_app_dirname** variable is set by the application to indicate the directory where application-specific Tcl files and packages are found. This is a read-only variable.

SEE ALSO

get_app_var(2)

sh_user_man_path

Indicates a directory root where you can store man pages for display with the **man** command.

TYPE

list

DEFAULT

empty list

DESCRIPTION

The **sh_user_man_path** variable is used to indicate a directory root where you can store man pages for display with the **man** command. The directory structure must start with a directory named *man*. Below *man* are directories named *cat1*, *cat2*, *cat3*, and so on. The **man** command will look in these directories for files named *file.1*, *file.2*, and *file.3*, respectively. These are pre-formatted files. It is up to you to format the files. The **man** command effectively just types the file.

These man pages could be for your Tcl procedures. The combination of defining help for your Tcl procedures with the **define_proc_attributes** command, and keeping a manual page for the same procedures allows you to fully document your application extensions.

The **man** command will look in **sh_user_man_path** after first looking in application-defined paths. The user-defined paths are consulted only if no matches are found in the application-defined paths.

To determine the current value of this variable, use the **get_app_var sh_user_man_path** command.

SEE ALSO

define_proc_attributes(2)
get_app_var(2)
man(2)
set_app_var(2)

svr_enable_vpp

Enables or disables preprocessing of Verilog files by the Verilog preprocessor.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

This variable is used only by the native Verilog reader. When set to *true*, before the Verilog reader reads a Verilog file, the Verilog preprocessor scans for and expands the Verilog preprocessor directives `define, `undef, `include, `ifdef, `else, and `endif. Intermediate files from the preprocessor are created in the directory referenced by the **gca_tmp_dir** variable. Also, the `include directive uses the **search_path** variable to find files.

Very few structural Verilog files use preprocessor directives. Set this variable to *true* only if your Verilog file contains directives that require the preprocessor. Without the preprocessor, the native Verilog reader does not recognize these directives.

To determine the current value of this variable, type **printvar svr_enable_vpp** or **echo $svr_enable_vpp**.

SEE ALSO

printvar(2)
read_verilog(2)
gca_tmp_dir(3)
search_path(3)

svr_keep_unconnected_cells

TYPE

Boolean

DEFAULT

true

DESCRIPTION

This variable is used only by the native Verilog reader. When set to *false*, if all instances of a certain reference are made without any connections, those cell instances are discarded. This saves memory by not representing cells which have no impact on the timing of the design, for instance metal fill cells. When set to *true*, such cells are preserved.

To determine the current value of this variable, type **printvar svr_keep_unconnected_cells** or **echo $svr_keep_unconnected_cells**.

SEE ALSO

printvar(2)
read_verilog(2)

svr_keep_unconnected_nets

TYPE

Boolean

DEFAULT

true

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

This variable is used only by the native Verilog reader. When *true* (the default), unconnected nets are preserved. When set to *false*, unconnected nets are discarded.

To determine the current value of this variable, type **printvar svr_keep_unconnected_nets** or **echo $svr_keep_unconnected_nets**.

SEE ALSO

printvar(2)
read_verilog(2)

synopsys_program_name

Indicates the name of the program currently running.

TYPE

string

DESCRIPTION

This variable is read only, and is set by the application to indicate the name of the program you are running. This is useful when writing scripts that are mostly common between some applications, but contain some differences based on the application.

To determine the current value of this variable, use **get_app_var synopsys_program_name**.

SEE ALSO

get_app_var(2)

synopsys_root

Indicates the root directory from which the application was run.

TYPE

string

DESCRIPTION

This variable is read only, and is set by the application to indicate the root directory from which the application was run.

To determine the current value of this variable, use **get_app_var synopsys_root**.

SEE ALSO

get_app_var(2)

timing_all_clocks_propagated

Determines whether or not all clocks are created as propagated clocks.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

When set to *true*, all clocks subsequently created by **create_clock** or **create_generated_clock** are created as propagated clocks. When set to *false* (the default), clocks are created as nonpropagated clocks.

By default, **create_clock** and **create_generated_clock** create only nonpropagated clocks. You can subsequently define some or all clocks to be propagated clocks using the **set_propagated_clock** command. However, if you set the **timing_all_clocks_propagated** variable to *true*, **create_clock** and **create_generated_clock** subsequently create only propagated clocks. Setting this variable to *true* or *false* affects only clocks created after the setting is changed. Clocks created before the setting is changed retain their original condition (propagated or nonpropagated).

To determine the current value of this variable, type **printvar timing_all_clocks_propagated** or **echo $timing_all_clocks_propagated.**

SEE ALSO

create_clock(2)
create_generated_clock(2)
printvar(2)
set_propagated_clock(2)

timing_arc_attributes

Lists the predefined attributes for timing arcs.

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Attributes are properties assigned to objects such as timing arcs, pins, cells, and nets. Definitions for timing_arc attributes are provided in the subsections that follow.

Attributes are informational, or "read-only." You cannot set the value of attributes designed by the application.

To determine the value of an attribute, use the **get_attribute** command. To see a list of all attributes available for a class of objects use the **list_attributes -application** command.

Collections of timing arcs are created with the command **get_timing_arcs**.

Timing Arc Attributes

from_pin

A collection attribute returning the from pin for a timing_arc. If this timing_arc is for a timing check such as setup or hold, the from pin is the clock pin of the check.

invalid_reason

A string attribute for the reason why the timing arc is not an active arc. See the description of is_disable and is_invalid attributes. The possible invalid_reason strings are:

set_disable_timing
auto_loop_breaking
inherited_loop_breaking
user_case_0
user_case_1
netlist_tie_0
netlist_tie_1
case_disabled
case_0
case_1
conditional_arc_disabled
default_arc_disabled
mode_values_disabled
pin_disabled
lib_arc_disabled
preset_clear_disabled

is_cellarc

A Boolean attribute that is true if this timing arc is for a cell. If this attribute is true both the from_pin and to_pin will be on the same cell.

is_db_inherited_disabled

A Boolean attribute that is true if the timing_arc is disabled because of a property from another tool.

is_disabled

A Boolean attribute for whether this timing arc is disabled. This attribute has the same definition as it does in PrimeTime. It is true only if the arc has been directly disabled by **set_disable_timing**, case propagation, loop breaking, and arc modes or conditions disabled. This attribute is true for a subset of the timing arcs that is_invalid attribute is true for. It does not include timing_arcs that

are disabled indirectly. Examples of timing_arcs that are disabled indirectly are timing arcs where the lib_timing_arc are disable or those with either the from or to pin disabled.

is_invalid

A Boolean attribute for whether this timing arc is not active. This attribute is true for a superset of the timing arcs that is_disable attribute is true for. See the description of the attribute "invalid_reason" for the possible reasons a timing_arc might be invalid.

is_user_disabled

A Boolean attribute for whether this timing arc has been disabled by you.

mode

A string attribute with the name of the mode this timing arc is active for. The attribute is only defined for moded arcs.

object_class

A string attribute with the value of "timing_arc". Each object class has a different object class value.

sense

A string attribute for the sense of a timing_arc. Possible values are

rising_edge rising_to_rise falling_to_rise
falling_edge rising_to_fall falling_to_fall
rise_to_rise fall_to_rise rise_to_fall
fall_to_fall positive_unate negative_unate
non_unate clear_high clear_low
clear_both preset_high preset_low
preset_both disable_high disable_low
disable_both disable_high_rise
disable_low_rise disable_both_rise
disable_high_fall disable_low_fall
disable_both_fall
enable_high enable_low enable_both
enable_high_rise enable_low_rise enable_both_rise
enable_high_fall enable_low_fall enable_both_fall
retain_rising_edge retain_rise_to_rise
retain_fall_to_rise retain_falling_edge
retain_rise_to_fall retain_fall_to_fall
retain_positive_unate retain_negative_unate
retain
max_clock_tree_positive_unate max_clock_tree_rise_to_rise
max_clock_tree_fall_to_fall max_clock_tree_negative_unate
max_clock_tree_rise_to_fall max_clock_tree_fall_to_rise
max_clock_tree_non_unate min_clock_tree_positive_unate
min_clock_tree_rise_to_rise min_clock_tree_fall_to_fall
min_clock_tree_negative_unate min_clock_tree_rise_to_fall
min_clock_tree_fall_to_rise min_clock_tree_non_unate
nonseq_setup_clk_rise nonseq_setup_rise_clk_rise
nonseq_setup_fall_clk_rise nonseq_setup_clk_fall
nonseq_setup_rise_clk_fall nonseq_setup_fall_clk_fall
nonseq_hold_clk_rise nonseq_hold_rise_clk_rise
nonseq_hold_fall_clk_rise nonseq_hold_clk_fall
nonseq_hold_rise_clk_fall nonseq_hold_fall_clk_fall
clock_gating_setup_clk_rise clock_gating_setup_rise_clk_rise
clock_gating_setup_fall_clk_rise clock_gating_setup_clk_fall
clock_gating_setup_rise_clk_fall clock_gating_setup_fall_clk_fall
clock_gating_hold_clk_rise clock_gating_hold_rise_clk_rise

to_pin

A collection attribute returning the to pin for a timing_arc. If this timing_arc is for a timing check such as setup or hold, the from pin is the data pin of the check.

user_clock_sense

A string attribute for the sense set by the command **set_clock_sense**. This attribute is only define on those pins where the **set_clock_sense** command was used directly. The possible values are: "positive", "negative", "rise_triggered_low_pulse", "rise_triggered_high_pulse", "fall_triggered_low_pulse", "fall_triggered_high_pulse".

when

A string attribute returning the when value for the timing_arc.

SEE ALSO

get_attribute(2)
list_attributes(2)

timing_arcs_include_inferred_checks

Controls whether constraint consistency include inferred checks in the timing arcs.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

When set to *true*, the **get_timing_arcs** command includes checks added by the tool that are not in the library models for the cells. For example the checks added for clock gating, can be retrieved by get_timing_arcs if this variable is set to *true*.

To determine the current value of this variable, type **printvar timing_arcs_include_inferred_checks**.

SEE ALSO

printvar(2)

timing_clock_gating_check_fanout_compatibility

Controls whether the effects of the **set_clock_gating_check** command propagates through logic, or applies only to the specified design object.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

This variable controls the behavior of the **set_clock_gating_check** command when it is applied to design netlist objects (ports, pins, or cells).

When this variable is set to *false*, constraint consistency uses the current behavior where the effects of the **set_clock_gating_check** command apply only to the specified design objects, and do not propagate through the transitive fanout. When this behavior is enabled, specifying the command on a port has no effect. This behavior is consistent with PrimeTime, Design Compiler, and IC Compiler.

To apply the clock gating settings to an entire clock domain without enumerating all gating cells, clock objects can be provided to the **set_clock_gating_check** command. When clock objects are provided, the clock gating settings apply to all instances of clock gating for those clocks.

When this variable is set to *true*, constraint consistency uses the behavior from older versions where the **set_clock_gating_check command** also applies to the transitive fanout of the specified design objects. There is no way to configure only the specified design objects without also propagating the clock gating settings to downstream logic.

To determine the current value of this variable, enter the following command:

```
ptc_shell> printvar timing_clock_gating_check_fanout_compatibility
```

SEE ALSO

set_clock_gating_check(2)

timing_clock_gating_propagate_enable

Allows the gating enable signal delay to propagate through the gating cell.

TYPE

int

DEFAULT

true

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

When set to *true* (the default), the tool allows the delay and slew from the data line of the gating check to propagate. When set to *false*, the tool blocks the delay and slew from the data line of the gating check from propagating. Only the delay and slew from the clock line is propagated.

If the output goes to a clock pin of a latch, setting this variable to *false* produces the most desirable behavior.

If the output goes to a data pin, setting this variable to *true* produces the most desirable behavior.

To determine the current value of this variable, type one of the following commands:

ptc_shell> **printvar timing_clock_gating_propagate_enable**

or

ptc_shell> **echo $timing_clock_gating_propagate_enable**

SEE ALSO

printvar(2)

timing_disable_clock_gating_checks

Disables checking for setup and hold clock gating violations.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

When set to *true*, this variable disables clock-gating setup and hold checks. When set to *false* (the default), the tool automatically determines clock-gating and performs clock-gating setup and hold checks.

To determine the current value of this variable, enter one of the following commands:

ptc_shell> **printvar timing_disable_clock_gating_checks**

or

ptc_shell> **echo $timing_disable_clock_gating_checks**

SEE ALSO

set_clock_gating_check(2)

timing_disable_cond_default_arcs

Disables the default, nonconditional timing arc between pins that do have conditional arcs.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

When *true*, disables nonconditional timing arcs between any pair of pins that have at least one conditional arc. When *false* (the default), these nonconditional timing arcs are not disabled. This variable is primarily intended to deal with the situation between two pins that have conditional arcs, where there is always a default timing arc with no condition.

Set this variable to *true* when the specified conditions cover all possible state-dependent delays, so that the default arc is useless. For example, consider a 2-input XOR gate with inputs as A and B and with output as Z. If the delays between A and Z are specified with 2 arcs with respective conditions 'B' and 'B~", the default arc between A and Z is useless and should be disabled.

To determine the current value of this variable, type one of the following commands:

```
ptc_shell> printvar timing_disable_cond_default_arcs
or

ptc_shell> echo $timing_disable_cond_default_arcs
```

SEE ALSO

report_disable_timing(2)

timing_disable_internal_inout_cell_paths

Enables bidirectional feedback paths within a cell.

TYPE

Boolean

DEFAULT

true

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

When set to *true* (the default), constraint consistency automatically disables bidirectional feedback paths in a cell. When set to *false*, bidirectional feedback paths in cells are enabled.

This variable has no effect on timing of bidirectional feedback paths that involve more than one cell (that is, if nets are involved); these feedback paths are controlled by the **timing_disable_internal_inout_net_arcs** variable.

To determine the current value of this variable, type one of the following commands:

ptc_shell> **printvar timing_disable_internal_inout_cell_paths**

or

ptc_shell> **echo $timing_disable_internal_inout_cell_paths**

SEE ALSO

remove_disable_timing(2)
set_disable_timing(2)
timing_disable_internal_inout_net_arcs(3)

timing_disable_internal_inout_net_arcs

Controls whether bidirectional feedback paths across nets are disabled or not.

TYPE

Boolean

DEFAULT

true

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

When set to *true* (the default), constraint consistency automatically disables bidirectional feedback paths that involve more than one cell; no path segmentation is required. Note that only the feedback net arc between non-bidirectional driver and load is disabled. When set to *false*, these bidirectional feedback paths are enabled.

This variable has no effect on timing of bidirectional feedback paths that are completely contained in one cell (that is, if nets are not involved); these feedback paths are controlled by the **timing_disable_internal_inout_cell_paths** variable.

To determine the current value of this variable, type one of the following commands:

```
ptc_shell> printvar timing_disable_internal_inout_net_arcs

or
ptc_shell> echo $timing_disable_internal_inout_net_arcs
```

SEE ALSO

remove_disable_timing(2)
set_disable_timing(2)
timing_disable_internal_inout_cell_paths(3)

timing_disable_recovery_removal_checks

Disables or enable the timing analysis of recovery and removal checks in the design.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

When set to *true*, this variable disables recovery and removal timing analysis. When set to *false* (the default), constraint consistency performs recovery and removal checks; for descriptions of these checks, see the man page for the **report_constraint** command.

To determine the current value of this variable, type one of the following commands:

ptc_shell> **printvar timing_disable_recovery_removal_checks**

or

ptc_shell> **echo $timing_disable_recovery_removal_checks**

SEE ALSO

timing_enable_auto_mux_clock_exclusivity

Enables automatic inference of MUX cells for clock exclusivity.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

If you set this variable to **true**, the tool automatically identifies the MUX cells on the clock network and forces the clocks that go through different data lines of the MUX cells to be exclusive.

By default, the variable is set to **false** and the tool does not automatically infer the exclusivity of clocks from MUX cells in the clock network. Please note that if there is a generated clock defined after the exclusivity point, the exclusivity states will be lost. This is similar to the existing behavior of the generated clocks. When a generated clock is created at a pin, all other clocks arriving at that pin are blocked unless they also have generated clock versions created at that pin.

Please note that this feature is not currently supported in **extract_model** and **write_eco_design** commands. PrimeTime will honor exclusivity settings in Hyperscale. Timing context with exclusivity will be captured by **characterize_context** command, and written out by **write_context** command. Please note that cross-boundary exclusivity will only be written in binary GBC format. It is not included in ASCII PTSH constraint format.

SEE ALSO

set_clock_exclusivity(2)
set_disable_auto_mux_clock_exclusivity(2)
report_clock(2)

timing_enable_clock_propagation_through_preset_clear

Enables propagation of clock signals through preset and clear pins

TYPE

Boolean

DEFAULT

false

GROUP

Timing variables

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

When this variable is set to *true*, clock signals are propagated through the preset and clear pins of a sequential device. Naturally, this only occurs when clock signals are incident on such pins.

If CRPR is enabled, it considers any sequential devices in the fanout of such pins for analysis.

Use the following command to determine the current value of the variable:

```
ptc_shell> printvar timing_enable_clock_propagation_through_preset_clear
```

SEE ALSO

timing_remove_clock_reconvergence_pessimism(2)

timing_enable_clock_propagation_through_three_state_enable_pins

Allows the clocks to propagate through the enable pin of a three-state cell.

TYPE

int

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

When set to *true*, constraint consistency allows the clocks to propagated through the enable pins of tristates. When set to *false* (the default), constraint consistency does not propagate clocks between a pair of pins if there is at least one timing arc with a disable sense between those pins.

To determine the current value of this variable, type one of the following commands:

ptc_shell> **printvar timing_enable_clock_propagation_through_three_state_enable_pins**

or

ptc_shell> **echo $timing_enable_clock_propagation_through_three_state_enable_pins**

SEE ALSO

timing_enable_multiple_clocks_per_reg

Enables or disables analysis of multiple clocks that reach a single register.

TYPE

Boolean

DEFAULT

true

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

This variable enables or disables analysis of multiple clocks that reach a register clock pin. When set to *true* (the default), all clocks reaching the register are analyzed simultaneously. When set to *false*, constraint consistency selects a random clock for analysis from among all clocks reaching a register clock pin. Do not change the value of **timing_enable_multiple_clocks_per_reg** from the default (*true*) unless you want this behavior.

If you set this variable to false and your design has multiple clocks per register, you should specify a clock to use with **set_data_check -clock**.

For the current value of this variable, type

printvar timing_enable_multiple_clocks_per_reg

SEE ALSO

create_clock(2)
create_generated_clock(2)
set_data_check(2)
printvar(2)

timing_enable_preset_clear_arcs

Controls whether constraint consistency enables or disables preset and clear arcs.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

When set to *true*, this variable permanently enables asynchronous preset and clear timing arcs, so that you use them to analyze timing paths. When set to *false* (the default), constraint consistency disables all preset and clear timing arcs.

Note that if there are any minimum pulse width checks defined on asynchronous preset and clear pins, they are performed regardless of the value of this variable. Also note the -true and the -justify options of report_timing cannot be used unless this variable is at its default.

To determine the current value of this variable, type

```
ptc_shell> printvar timing_enable_preset_clear_arcs
```

SEE ALSO

printvar(2)
report_timing(2)

timing_gclock_source_network_num_master_registers

Allows you to control the maximum number of register clock pins clocked by the master clock allowed in generated clock source latency paths.

TYPE

int

DEFAULT

10000000

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

This variable allows you to control the maximum number of register clock pins clocked by the master clock allowed in generated clock source latency paths. The variable does not effect the number of register traversed in a single path that do not have a clock assigned or are clocked by another generated clock that has the same primary master as the generated clock in question.

Register clock pins or transparent-D pins of registers clocked by unrelated clocks are not traversed in determining generated clock source latency paths. An unrelated clock is any clock that primary master clock differs from the generated clock who source latency paths are being computed.

To determine the current value of this variable, type one of the following commands:

ptc_shell> **printvar timing_gclock_source_network_num_master_registers**

or

ptc_shell> **echo $timing_gclock_source_network_num_master_registers**

SEE ALSO

timing_ideal_clock_zero_default_transition

Specifies whether or not a zero transition value is assumed for sequential devices clocked by ideal clocks.

TYPE

Boolean

DEFAULT

true

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Specifies a transition value to use at clock pins of a flip-flop. If the **set_clock_transition** command is used and the clock is ideal, the transition value is used. This variable has no effect.

If the clock transition is not set by **set_clock_transition** command, and the clock is ideal, then this variable will have the following effect. When set to *true* (the default), constraint consistency uses a zero transition value for ideal clocks. When set to *false*, constraint consistency uses a propagated transition value. Note that set_clock_transition overrides the effect of this variable, when the clock is ideal.

This behavior differs from the previous behavior, where constraint consistency used a propagated transition value for an ideal clock, but zero delay values at the clock pins.

To determine the current value of this variable, type one of the following commands:

ptc_shell> **printvar timing_ideal_clock_zero_default_transition**

or

ptc_shell> **echo $timing_ideal_clock_zero_default_transition**

SEE ALSO

set_clock_transition(2)

timing_input_port_clock_shift_one_cycle

Determines if paths originating at input ports are given an extra cycle to meet their timing constraints.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Affects the behavior of constraint consistency when timing a path from an input port with no clocked input external delay. When set to *true*, paths starting at such input ports are given one extra cycle (set_multicycle_path 2) to meet timing constraints at clocked destination registers or output ports. When set to *false* (the default), no extra multicycle shift is applied.

To determine the current value of this variable, type

printvar timing_input_port_clock_shift_one_cycle

SEE ALSO

report_timing(2)

timing_input_port_default_clock

Determines whether a default clock is assumed at input ports for which you have not defined a clock with **set_input_delay**.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

This Boolean variable affects the behavior of constraint consistency when you set an input delay without a clock on an input port. When set to *true* (the default), the input delay on the port is set with respect to one imaginary clock so that the inputs are constrained. This also causes the clocks along the paths driven by these input ports to become related. Moreover, the period of this clock is equal to the base period of all these related clocks. When set to *false*, no such imaginary clock is assumed.

To determine the current value of this variable, type

```
ptc_shell> printvar timing_input_port_default_clock
```

SEE ALSO

set_input_delay(2)

user_units_from_first_library

Specifies whether to set the input and output units to those found in the first cell library in the **link_path**.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

This command is available only if you invoke the pt_shell with the **-constraints** option.

Tools such as Design Compiler generally determine input and output units from the units in the first cell library in the **link_path**. Setting this variable to *true* makes ptc_shell more compatible with such flows.

It is recommended to use a value of *false* (the default) and explicitly set input and output units.

SEE ALSO

set_input_units(2)
set_output_units(2)