Synthesis Variables and Attributes

Version W-2024.09-SP3, January 2025

SYNOPSYS®

Feedback

Contents

1

Synthesis Variables

This document describes the variables supported by the Design Compiler tool.

D

DW_lp_op_iso_mode

Specify the datapath gating style for instances of DesignWare low power components that have op_iso_mode parameters.

Data Types

list

Default NONE

Description

This variable specifies the datapath gating style inside a low power DesignWare component that has 'op_iso_mode' parameter.

By default, components that have a 'op_iso_mode' parameter have its value set to '0' which allows the 'DW_lp_op_iso_mode' variable to control the datapath gating style.

There are 4 possible values for DW_lp_op_iso_mode: NONE, adaptive, AND, OR. When the value is set to 'NONE', there is no datapath gating happing in the design; When the value is set to 'adaptive', preferred gating style. 'and' or 'or' depending on component When the value is set to 'AND', gating style is 'and'; When the value is set to 'OR', gating style is 'or';

Setting the 'op_iso_mode' parameter to a value other than the default value of '0' (zero) on an instance allows the RTL source code to override the the value set by the 'DW_lp_op_iso_mode' variable.

The op_iso_mode parameter inside a DesignWare component has 5 possible values: 0: apply Design Compiler variable 'DW_lp_op_iso_mode' (default value) 1: 'none' (overrides 'DW_lp_op_iso_mode' setting) 2: 'and' (overrides 'DW_lp_op_iso_mode' setting) 3: 'or' (overrides 'DW_lp_op_iso_mode' setting) 4: preferred gating style. 'and' or 'or' depending on component (overrides 'DW_lp_op_iso_mode' setting)

Feedback

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

```
prompt> printvar DW_lp_op_iso_mode
```

M

MasterInstance

Data Types

string

Description

If a design has multiple instances, one way to resolve the multiple instances before compiling (when using Automated Chip Synthesis) is to select a master instance. Selecting a master instance is similar to using the *compile-dont_touch* method of resolving instances. The master instance is compiled and used for all of the instances referencing that design.

Use the *sub_designs_of* command to determine if your design contains multiple instances of any subdesign. Use the *sub_instances_of* command to identify the multiple instances.

Use the *set_attribute* command to set the *MasterInstance* attribute on the instance you want to use as the master instance.

You can also resolve multiple instances by uniquifying or ungrouping.

See Also

- sub_designs_of
- sub_instances_of
- ungroup
- uniquify

a

abstraction_enable_power_calculation

Perform power calculations on a design that is to be used as an abstract block.

Feedback

Data Types

Boolean

Default true

Description

The *create_block_abstraction* command processes the block, creates the abstraction information and annotates it on the design in memory. By default, *create_block_abstraction* command invoke the power calculation during it's operation. Subsequent power reporting commands at top level can report power consumption correctly.

Power consumption of the block is calculated during block abstraction. The power data are stored as attributes on the abstract block and they are used by the *report_power* command during the final assembly step of the entire chip. Thus, the *report_power* command is able to report more accurate power consumption for the designs with abstract blocks.

The *abstraction_enable_power_calculation* variable instructs the *create_block_abstraction* command to enable or disable the power calculation.

The *abstraction_enable_power_calculation* variable is on (*true*) by default. If a power license is not available, the power calculation step is disrupted.

Setting this variable as false does not check for power license availability and power calculation step is not invoked.

See Also

- create_block_abstraction
- report_power

---

abstraction_ignore_percentage

Specifies a threshold for the percentage of total registers in the transitive fanout of an input port, beyond which the port is to be ignored when identifying interface logic.

Data Types

float

Default 25

Description

The value you specify for the *abstraction_ignore_percentage* variable defines a minimum threshold for the percentage of the total registers in the transitive fanout of an input port,

Feedback

beyond which the tool is to ignore the port when identifying interface logic. The default is 25.

The tool automatically ignores ports if the ports fan out to a percentage of total registers in the design that is greater than or equal to the value you specify for this variable.

For a discussion about creating block abstractions and related commands, see the man page for the *create_block_abstraction* command.

To see the current value of this variable, type

prompt> *printvar abstraction_ignore_percentage*

See Also

- create_block_abstraction

---

access_internal_pins

Controls access of internal pins.

Data Types

Boolean

Default false

Group

system_variables

Description

The *access_internal_pins* variable controls your access to internal pins. Internal pins are related to the internal design inside a library cell.

When the *access_internal_pins* variable is set to true, you can query internal pins using the *get_pins* command and set attributes or constraints on the internal pins.

See Also

- get_pins

---

acs_area_report_suffix

Specifies the suffix for area reports generated during the automated compile process.

Data Types

string

Feedback

Default area in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

Specifies the suffix for area reports generated during the automated compile process. For use in dc_shell-t (Tcl mode of dc_shell) only. The default is *area*.

Unless you specify the *-no_reports* option, the compile scripts generated by the *write_compile_script* command include a command to generate an area report. The area report is placed in a file called pass*n*/reports/*design*.*suffix*, where *suffix* is the value of the *acs_area_report_suffix* variable.

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

---

acs_budgeted_cstr_suffix

Specifies the suffix for constraint files generated by the *derive_partition_budgets* command.

Data Types

string

Default con in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

Specifies the suffix for constraint files generated by the *derive_partition_budgets* command. For use in dc_shell-t (Tcl mode of dc_shell) only.

The *derive_partition_budgets* command generates a constraint file for each compile partition. The generated constraint files are named pass*n*/constraints/*design*.*suffix*, where *suffix* is the value of the *acs_budgeted_cstr_suffix* variable.

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

---

acs_compile_script_suffix

Specifies the default suffix for script files generated by the *write_compile_script* command, sourced in the makefile generated by the *write_makefile* command, and located by the *report_pass_data* command.

Feedback

Data Types

string

Default autoscr

Description

Specifies the default suffix for script files generated by the *write_compile_script* command, sourced in the makefile generated by the *write_makefile* command, and located by the *report_pass_data* command. For use in dc_shell-t (Tcl mode of dc_shell) only.

The script is placed in a file called pass*n*/scripts/*design*.*suffix*, where *suffix* is the value of the *acs_compile_script_suffix* variable.

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

---

acs_constraint_file_suffix

Specifies the default suffix for constraint files generated by *write_partition_constraints* during the automated compile process.

Data Types

string

Default con in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

Specifies the default suffix for constraint files generated by *write_partition_constraints* during the automated compile process. For use in dc_shell-t (Tcl mode of dc_shell) only.

The constraints are placed in a file called pass*n*/constraints/*design*.*suffix*, where *suffix* is the value of the *acs_constraint_file_suffix* variable.

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

---

acs_cstr_report_suffix

Specifies the default suffix for constraint reports generated during the automated compile process.

Data Types

string

Feedback

Default cstr in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Specifies the default suffix for constraint reports generated during the automated compile process. For use in dc_shell-t (Tcl mode of dc_shell) only.

The constraint report is placed in a file called pass*n*/reports/*design*.*suffix*, where *suffix* is the value of the *acs_cstr_report_suffix* variable.

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

See Also

- compile

---

acs_db_suffix

Specifies the default suffix for .db files that are read or written during the automated compile process.

Data Types

string

Default db in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Specifies the default suffix for .db files that are read or written during the automated compile process. For use in dc_shell-t (Tcl mode of dc_shell) only.

The .db is placed in files called pass*n*/db/pre_compile/*design*.*suffix* and pass*n*/db/post_compile/*design*.*suffix*, where *suffix* is the value of the *acs_db_suffix* variable.

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

---

acs_global_user_compile_strategy_script

Specifies the base file name for the user-defined default compile strategy.

Feedback

Data Types

string

Default default.compile in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Specifies the base file name for the user-defined default compile strategy. For use in dc_shell-t (Tcl mode of dc_shell) only. A compile strategy is a script that includes commands to set the *compile* variables, set the *compile* attributes, and run the *compile* command.

Before generating a partition compile script, Automated Chip Synthesis looks for a user-defined partition compile strategy in the file scripts/pass*n*/*partition_name*.*suffix*. If a partition-specific compile strategy does not exist, Automated Chip Synthesis looks for a user-defined default compile strategy in the file scripts/pass*n*/$acs_global_user_compile_strategy_script.*suffix*. If a user-defined compile strategy exists, Automated Chip Synthesis uses it instead of the default compile strategy when generating the compile script.

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

See Also

- acs_user_compile_strategy_script_suffix

---

acs_log_file_suffix

Specifies the default suffix for log files generated during the automated compile process.

Data Types

string

Default log in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Feedback

Description

Specifies the default suffix for log files generated during the automated compile process. For use in dc_shell-t (Tcl mode of dc_shell) only.

The log is placed in a file called pass*n*/logs/*design*.*suffix*, where *suffix* is the value of the *acs_log_file_suffix* variable.

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

---

acs_makefile_name

Specifies the file name for the makefile generated by the *write_makefile* command and run by the *compile_partitions* command.

Data Types

string

Default Makefile

Description

Specifies the filename for the makefile generated by the *write_makefile* command and run by the *compile_partitions* command. For use in dc_shell-t (Tcl mode of dc_shell) only.

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

---

acs_override_script_suffix

Specifies the suffix for user-defined partition compile scripts.

Data Types

string

Default uscr in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Specifies the suffix for user-defined partition compile scripts. For use in dc_shell-t (Tcl mode of dc_shell) only.

Before generating a partition compile script, Automated Chip Synthesis looks for a user-defined partition compile script in scripts/pass*n*/*partition_name*.*suffix*. If a user-defined script exists, Automated Chip Synthesis uses it instead of generating a default script.

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

---

acs_qor_report_suffix

Specifies the suffix for QOR reports generated during the automated compile process; the default is *qor*.

Data Types

string

Default qor in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Specifies the suffix for QOR reports generated during the automated compile process; the default is *qor*. For use in dc_shell-t (Tcl mode of dc_shell) only.

Unless you specify the *-no_reports option*, the compile scripts generated by the *write_compile_script* command includes a command to generate a QOR report. The QOR report is placed in a file called pass*n*/reports/*design*.*suffix*, where *suffix* is the value of the *acs_qor_report_suffix* variable.

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

---

acs_timing_report_suffix

Specifies the suffix for timing reports generated during the automated compile process.

Data Types

string

Default tim in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Specifies the suffix for timing reports generated during the automated compile process. For use in dc_shell-t (Tcl mode of dc_shell) only.

Feedback

Unless you specify the *-no_reports* option, the compile scripts generated by the *write_compile_script* command include a command to generate a timing report. The timing report is placed in a file called pass*n*/reports/*design*.*suffix*, where *suffix* is the value of the *acs_timing_report_suffix* variable.

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

---

acs_user_compile_strategy_script_suffix

Specifies the suffix for user-defined partition compile strategies.

Data Types

string

Default compile

Description

Specifies the suffix for user-defined partition compile strategies. For use in dc_shell-t (Tcl mode of dc_shell) only.

A compile strategy is a script that includes commands to set the *compile* variables, set the *compile* attributes, and run the *compile* command.

Before generating a partition compile script, Automated Chip Synthesis looks for a user-defined partition compile strategy in scripts/pass*n*/*partition_name*.*suffix*. If a user-defined compile strategy exists, Automated Chip Synthesis uses it when generating the compile script, instead of using the default compile strategy.

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

---

acs_work_dir

Specifies the root of the Automated Chip Synthesis project directory.

Data Types

string

Default the current working directory

Description

Specifies the root of the Automated Chip Synthesis project directory. During startup, this variable is set to the current working directory. For use in dc_shell-t (Tcl mode of dc_shell) only.

Feedback

This value is used when locating input files for Automated Chip Synthesis commands and when generating absolute path names.

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

Note: Character @ is reserved to specify pass dependent directory names. Its use as a literal character in ACS directory paths is not allowed.

---

alib_library_analysis_path

Specifies a single path, similar to a search path, for reading and writing the alib files that correspond to the target libraries.

Data Types

string

Default "./"

Description

This variable specifies the path from which the tool loads alibs during compile. The tool stores alibs to this path when the *alib_analyze_libs* command is issued.

See Also

- alib_analyze_libs

---

all_registers_include_icg

Enables command all_registers to include integrated clock-gating cells.

Data Types

Boolean

Default false

Description

When set to *true*, this variable enables command all_registers to include integrated clock-gating cells.

To determine the current setting, use the following command:

```
get_app_var all_registers_include_icg
```

Feedback

See Also

- all_registers

---

attributes

Lists the predefined Synopsys attributes.

Description

Attributes are properties assigned to objects such as nets, cells, and clocks, and describe design features to be considered during optimization.

Attributes are grouped into the following categories:

- cell
- clock
- design
- library cell
- net
- pin
- port
- read-only
- reference

Definitions for these attributes are provided in the subsections that follow.

There are several commands used to set attributes; however, most attributes can be set with the *set_attribute* command. If the attribute definition specifies a set command, use it to set the attribute. Otherwise, use the *set_attribute* command.

Some attributes are informational, or read-only. You cannot set the value of these attributes. Most attribute groups contain read-only attributes; however, a complete list of these attributes is provided in the "Read Only" subsection.

Some attributes are instance-specific, which means they can be applied to specified objects in the design hierarchy. The following attributes are instance-specific:

- disable_timing
- load
- test_assume

Certain attributes are specific to Power Compiler objects. For information about the Power Compiler attributes, see the *power_attributes* man page.

To determine the value of an attribute, use the *get_attribute* command. To remove attributes, use the *remove_attribute* command.

For a more detailed explanation of the attributes, see the man pages of the appropriate set command.

Note that path groups, cell delay, net delay, external delay, point-to-point timing specification, and arrival information are not represented as attributes, and therefore cannot be manipulated with attribute commands.

Cell Attributes

async_set_reset_q

Establishes the value (0 or 1) that should be assigned to the q output of an inferred register if set and reset are both active at the same time. To be used with the *async_set_reset_qn* attribute. Use these attributes if one of the following condition sets apply:

- You have used the *one_hot* or *one_cold* attribute or directive in your HDL description and your logic library is written using pre-V3.0a syntax
- Your logic library does not use a consistent convention for q and qn when set and reset are both active

By default, if set and reset are both active at the same time, Design Compiler uses the convention of the selected logic library, as set with the *target_library* variable. Set this attribute with the *set_attribute* command.

async_set_reset_qn

Establishes the value (0 or 1) that should be assigned to the qn output of an inferred register if set and reset are both active at the same time. To be used with the *async_set_reset_q* attribute. Use these attributes if one of the following condition sets apply:

- You have used the *one_hot* or *one_cold* attribute or directive in your HDL description and your logic library is written using pre-V3.0a syntax
- Your logic library does not use a consistent convention for q and qn when set and reset are both active

If a V3.0a or later syntax logic library is used, then by default, if set and reset are both active at the same time, Design Compiler will use the convention of the selected logic library (*target_library*). Set with *set_attribute*.

If you are unsure whether or not your logic library uses V3.0a syntax, ask your ASIC vendor.

combinational_type_exact

Specifies the replacement gate to use for cells specified in the cell list. Compile attempts to convert combinational gates tagged with *set_compile_type* to the specified replacement combinational gate. Set with *set_combinational_type*.

disable_timing

Disables the timing arcs of a cell. This has the same effect on timing as not having the arc in the library. Set with *set_disable_timing*.

dont_touch

Identifies cells to be excluded from optimization. Values are true (the default) or false. Cells with the *dont_touch* attribute set to true are not modified or replaced during compile. Setting *dont_touch* on a hierarchical cell sets the attribute on all cells below the hierarchical cell. Set with the *set_dont_touch command.*

flip_flop_type

Stores the name of the specified flip-flop to be converted from the *target_library*. The *compile* command automatically converts all tagged flip-flops to the specified (or one similar) type. Set with *set_register_type -flip_flop flip_flop_name [cell_list]* .

flip_flop_type_exact

Stores the name of the specified flip-flop to be converted from the *target_library*. The *compile* command automatically converts all tagged flip-flops to the exact flip-flop type. Set with *set_register_type -exact -flip_flop flip_flop_name [cell_list]* .

is_black_box

Sets to true if the cell's reference is not linked to a design or is linked to a design that does not have a functionality.

This attribute is read-only and cannot be set by the user.

is_combinational

Sets to true if all cells of a design and all designs in its hierarchy are combinational. A cell is combinational if it is non-sequential or non-tristate and all of its outputs compute a combinational logic function. The *report_lib* command reports such a cell as not a black-box.

This attribute is read-only and cannot be set by the user.

is_dw_subblock

Sets to true if the object (a cell, a reference, or a design) is a DW subblock that was automatically elaborated.

This attribute is read-only and cannot be set by the user.

Note that DW subblocks that are manually elaborated do not have this attribute.

is_hierarchical

Sets to true if the design is not a leaf design; for example, not from a logic library.

This attribute is read-only and cannot be set by the user.

is_mapped

Sets to true if the cell is not generic logic.

This attribute is read-only and cannot be set by the user.

is_sequential

Sets to true if the cell is sequential. A cell is sequential if it is not combinational.

This attribute is read-only and cannot be set by the user.

is_synlib_module

Sets to true if the object (a cell, a reference, or a design) refers to an unmapped module reference, or if the object is (or refers to) a design that was automatically elaborated from a synlib module or a synlib operator.

This attribute is read-only and cannot be set by the user.

Note that synlib modules that are manually elaborated do not have this attribute.

is_synlib_operator

Sets to true if the object (a cell or a reference) is a synthetic library operator reference.

This attribute is read-only and cannot be set by the user.

is_test_circuitry

Sets by *insert_dft* on the scan cells and nets added to a design during the addition of test circuitry.

This attribute is read-only and cannot be set by the user.

is_unmapped

Sets to true if the cell is generic logic.

This attribute is read-only and cannot be set by the user.

latch_type_exact

Stores the name of the specified latch to be converted from the *target_library*. The *compile* command automatically converts all tagged latches to the exact latch type. Set with *set_register_type -latch latch_name [cell_list]* .

map_only

Specifies that *compile* attempts to map the object exactly in the target library, and exclude the object from logic-level optimization (flattening and structuring) when set to true. The default is false. Set with *set_map_only*.

max_fall_delay

Specifies a floating-point value that establishes the maximum falling delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_max_delay*.

max_rise_delay

Specifies a floating-point value that establishes the maximum rising delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_max_delay*.

max_time_borrow

Specifies a floating-point number that establishes an upper limit for time borrowing; that is, it prevents the use of the entire pulse width for level-sensitive latches. Units are those used in the logic library. Set with *set_max_time_borrow*.

min_fall_delay

Specifies a floating-point value that establishes the minimum falling delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_min_delay*.

min_rise_delay

Specifies a floating-point value that establishes the minimum rising delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_min_delay*.

ref_name

Specifies the reference name of a cell.

This attribute is read-only and cannot be set by the user.

full_name

Specifies the hierarchical name of cell, pin or net.

This attribute is read-only and cannot be set by the user.

scan

Specifies that the cell is always replaced by an equivalent scan cell during *insert_dft* when set to true. When set to false, the cell is not replaced. Set with *set_scan*.

scan_chain

Includes the specified cells of the referenced design in the scan-chain whose index is the value of this attribute. Set with *set_scan_chain*.

scan_element

Determines if sequential cells in the specified designs are replaced by equivalent scan cells during *insert_scan*. When true, the default, *insert_scan* replaces cell_design_ref_list with equivalent scan cells. The scan cells are not replaced when set to false. Set with *set_scan_element*.

scan_latch_transparent

Makes specified cells transparent during ATPG when set to true. For hierarchical cells, the effects apply hierarchically to level-sensitive leave cells. Set with *set_scan_transparent*. Remove with *remove_attribute*.

test_isolate

Indicates that the specified sequential cells, pins, or ports are to be logically isolated and considered untestable during test design rule checking by *check_test*. When this attribute is set on a cell, it is also placed on all pins of that cell. Do not set this attribute on a hierarchical cell. Use *report_test -assertions* for a report on isolated objects. Set with *set_test_isolate*.

Note that setting this attribute suppresses the warning messages associated with the isolated objects.

test_routing_position

Specifies the preferred routing order of the scan-test signals of the identified cells. Set with *set_test_routing_order*.

ungroup

Removes a level of hierarchy by exploding the contents of the specified cell in the current design. If specified on a reference object, cells using that reference are ungrouped during *compile*. Set with *set_ungroup*.

register_list

Specifies the single-bit cells that are remapped to the multibit cell. This attribute can be set when the compile_ultra or the create_regsiter_bank commands remap single-bit registers to multibit registers.

Feedback

Clock Attributes

clock_fall_transition

Sets the falling transition value on the specified clock list. The clock_fall_transition overrides the calculated transition times on clock pins of registers and associated nets. Set using *set_clock_transition*.

clock_rise_transition

Sets the rising transition value on the specified clock list. The clock_rise_transition overrides the calculated transition times on clock pins of registers and associated nets. Set using *set_clock_transition*.

dont_touch_network

When a design is optimized, *compile* assigns *dont_touch* attributes to all cells and nets in the transitive fanout of *dont_touch_network* ports. The *dont_touch* assignment stops at the boundary of storage elements. An element is recognized as storage only if it has setup or hold constraints. Set with *set_dont_touch_network.*

fix_hold

Specifies that *compile* should attempt to fix hold violations for timing endpoints related to this clock. Set with *set_fix_hold*.

hold_uncertainty

Specifies a negative uncertainty from the edges of the ideal clock waveform. Set with *set_clock_uncertainty -hold*.

max_fall_delay

A floating point value that specifies the maximum falling delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_max_delay*.

max_rise_delay

A floating point value that specifies the maximum rising delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_max_delay*.

max_time_borrow

A floating point number that establishes an upper limit for time borrowing; that is, it prevents the use of the entire pulse width for level-sensitive latches. Units are those used in the logic library. Set with *set_max_time_borrow*.

min_fall_delay

A floating point value that specifies the minimum falling delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_min_delay*.

min_rise_delay

A floating point value that specifies the minimum rising delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_min_delay*.

period

Assigns a value to the clock period. The clock period (or cycle time) is the shortest time during which the clock waveform repeats. For a simple waveform with one rising and one falling edge, the period is the difference between successive rising edges. Set with *create_clock -period_value*.

propagated_clock

Specifies that the clock edge times be delayed by propagating the values through the clock network. If this attribute is not present, ideal clocking is assumed. Set with *set_propagated_clock*.

setup_uncertainty

Specifies a positive uncertainty from the edges of the ideal clock waveform. Set with *set_clock_uncertainty -setup*.

Design Attributes

async_set_reset_q

Establishes the value (0 or 1) that should be assigned to the q output of an inferred register if set and reset are both active at the same time. To be used with *async_set_reset_qn*. Use these attributes only if you have used the *one_hot* or *one_cold* attributes/directives in your HDL description *and* your logic library is written using pre-V3.0a syntax; *or* if your logic library does not use a consistent convention for q and qn when set and reset are both active. If a V3.0a or later syntax logic library is used, then by default if set and reset are both active at the same time Design Compiler will use the convention of the selected logic library (*target_library*). Set with *set_attribute*.

Note: If you are unsure whether or not your logic library uses V3.0a syntax, ask your ASIC vendor.

async_set_reset_qn

Establishes the value (0 or 1) that should be assigned to the qn output of an inferred register if set and reset are both active at the same time. To be used with *async_set_reset_q*. Use these attributes only if you have used the *one_hot* or *one_cold* attributes/directives in your HDL description *and* your logic library is written using pre-V3.0a syntax; *or* if your logic library does not use a consistent convention for q and qn when set and reset are both active. If a V3.0a or later syntax logic library is used, then by default if set and reset are both active at the

same time Design Compiler will use the convention of the selected logic library (*target_library*). Set with *set_attribute*.

If you are unsure whether or not your logic library uses V3.0a syntax, ask your ASIC vendor.

balance_registers

Determines whether the registers in a design are retimed during *compile*. When *true* (the default value), *compile* invokes the *balance_registers* command, which moves registers to minimize the maximum register-to-register delay. Set this attribute to *false*, or remove it, to disable this behavior.

Set with *set_balance_registers*.

If your design contains generic logic, you should ensure that all components are mapped to cells from the library before setting the *balance_registers* attribute.

boundary_optimization

Enables *compile* to optimize across hierarchical boundaries. Hierarchy is ignored during optimization for designs with this attribute set to *true*. Set with *set_boundary_optimization* .

default_flip_flop_type

Specifies the default flip-flop type for the current design. During the mapping process, *compile* tries to convert all unmapped flip-flops to this type. If *compile* is unable to use this flip-flop, it maps these cells into the smallest flip-flop possible. Set with *set_register_type -flip_flop flip_flop_name*.

default_flip_flop_type_exact

During the mapping process, *compile* converts unmapped flip-flops to the exact flip-flop type specified here. Set with *set_register_type -exact -flip_flop flip_flop_name*

default_latch_type_exact

Specifies the exact default latch type for the *current_design*. During the mapping process, *compile* converts unmapped latches to the exact latch type specified here. Set with *set_register_type -exact -latch latch_name*.

design_type

Indicates the current state of the design and has the value *fsm* (finite state machine), *pla* (programmable logic array), *equation* (Boolean logic), or *netlist* (gates). This attribute is "read-only" and cannot be set by the user.

dont_touch

Identifies designs that are to be excluded from optimization. Values are *true* (the default) or *false*. Designs with the *dont_touch* attribute set to *true* are not modified or replaced during *compile*. Setting *dont_touch* on a design has an effect only when the design is instantiated within another design as a level of hierarchy; setting *dont_touch* on the top-level design has no effect. Set with the *set_dont_touch* command.

flatten

When set to *true*, determines that a design is to be flattened during *compile*. By default, a design is not flattened. Set with the *set_flatten* command.

flatten_effort

Defines the level of CPU effort that *compile* uses to flatten a design. Allowed values are *low* (the default), *medium*, or *high*. Set with the *set_flatten* command.

flatten_minimize

Defines the minimization strategy used for logic equations. Allowed values are *single_output*, *multiple_output*, or *none*. Set with the *set_flatten* command.

flatten_phase

When *true*, allows logic flattening to invert the phase of outputs during *compile*. By default, logic flattening does not invert the phase of outputs. Used only if the *flatten* attribute is set. Set with *set_flatten.*

implementation

The implementation for each specified instance of the specified component_type. Specifying the *-default* option removes this attribute from all instances of the component type in the current design. Set with *set_jtag_implementation*.

is_combinational

true if all cells of a design and all designs in its hierarchy are combinational. A cell is combinational if it is non-sequential or nonthree-state and all of its outputs compute a combinational logic function. The *report_lib* command will report such a cell as not a black-box. This attribute is read-only; you cannot set it.

is_dw_subblock

true if the object (a cell, a reference, or a design) is a DW subblock that was automatically elaborated. This attribute is "read-only" and cannot be set by the user.

NOTE: DW subblocks that are manually elaborated will not have this attribute.

is_hierarchical

true if any of the cells of a design are not leaf cells (for example, not from a logic library). This attribute is read-only and cannot be set by the user.

is_mapped

true if all the non-hierarchical cells of a design are mapped to cells in a logic library. This attribute is "read-only" and cannot be set by the user.

is_sequential

true if any cells of a design or designs in its hierarchy are sequential. A cell is sequential if it is not combinational (if any of its outputs depend on previous inputs). This attribute is read-only and cannot be set by the user.

is_synlib_module

true if the object (a cell, a reference, or a design) refers to an unmapped module reference or if the object is (or refers to) a design that was automatically elaborated from a synlib module or a synlib operator. This attribute is "read-only" and cannot be set by the user.

NOTE: synlib modules that are manually elaborated will not have this attribute.

is_unmapped

true if any of the cells are not linked to a design or mapped to a logic library. This attribute is read-only and cannot be set by the user.

local_link_library

A string that contains a list of design files and libraries to be added to the beginning of the *link_library* whenever a *link* operation is performed. Set with *set_local_link_library*.

map_only

When set to *true*, *compile* will attempt to map the object exactly in the target library, and will exclude the object from logic-level optimization (flattening and structuring). The default is *false*. Set with *set_map_only*.

max_area

A floating point number that represents the target area of the design. *compile* uses it to calculate the area cost of the design. The units must be consistent with the units used from the logic library during optimization. Set with *set_max_area*.

max_capacitance

A floating point number that sets the maximum capacitance value for input, output, or bidirectional ports, or designs. The units must be consistent with those of the logic library used during optimization. Set with *set_max_capacitance*.

max_total_power

A floating point number that specifies the maximum target total power for the *current_design*. Total power is defined as the sum of dynamic and leakage power. If this attribute is specified more than once for a design, the latest value is used. Set with *set_max_total_power*.

minimize_tree_delay

When *true* (the default value), *compile* restructures expression trees in the *current_design* or in a list of specified designs, to minimize tree delay. The value of this attribute overrides the value of *hlo_minimize_tree_delay*. Set this attribute to *false* for any designs that you do not wish to be restructured. Set with *set_minimize_tree_delay*.

model_map_effort

Specifies the relative amount of CPU time to be used by *compile* during modeling, typically for synthetic library implementations. Values are *low*, *medium*, and *high*, or 1, 2, and 3. If *model_map_effort* is not set, the value of *synlib_model_map_effort* is used. Set with the *set_model_map_effort* command.

model_scale

A floating point number that sets the model scale factor for the *current_design*. Set with *set_model_scale*.

optimize_registers

When *true* (the default value), *compile* automatically invokes the Behavioral Compiler *optimize_registers* command to retime the design during optimization. Setting the attribute to *false* disables this behavior. Your design cannot contain generic logic at the instant *optimize_registers* is invoked during *compile*.

Set with *set_optimize_registers*.

part

A string value that specifies the Xilinx part type for a design. For valid part types, refer to the Xilinx *XC4000 Databook*. Set with *set_attribute*.

port_is_pad

Indicates specified ports are to have I/O pads attached. The I/O pads are added during insert_pads and automatically added during compile. Set using *set_port_is_pad*.

resource_allocation

Indicates the type of resource allocation to be used by *compile* for the *current_design*. Allowed values are *none*, indicating no resource sharing; *area_only*, indicating resource sharing with tree balancing without considering

timing constraints; *area_no_tree_balancing*, indicating resource sharing without tree balancing and without considering timing constraints; and *constraint_driven* (the default), indicating resource sharing so that timing constraints are met or not worsened. The value of this attribute overrides the value of the variable *hlo_resource_allocation* for the *current_design*. Set with *set_resource_allocation*.

resource_implementation

Indicates the type of resource implementation to be used by *compile* for the *current_design*. Allowed values are *area_only*, indicating resource implementation without considering timing constraints; *constraint_driven*, indicating resource implementation so that timing constraints are met or not worsened; and *use_fastest*, indicating resource implementation using the fastest implementation initially, unless all timing constraints are met. If the fastest implementation has been selected initially later steps of the compile command will select components with smaller area later in uncritical parts of the design. The value of this attribute overrides the value of the variable *hlo_resource_implementation* for the *current_design*. Set with *set_resource_implementation*.

scan_element

Determines if sequential cells in the specified designs are replaced by equivalent scan cells or designs during *insert_scan*. Default is set to *true*. When set to *false*, sequential cells are not replaced by equivalent scan cells. Set with *set_scan_element*.

scan_latch_transparent

When set to *true*, makes specified designs transparent during ATPG. For hierarchical cells, the effects apply hierarchically to level-sensitive leaf cells. The *set_scan_transparent* command sets the attribute; the *remove_attribute* command removes it.

share_cse

When *true*, the value of the environment variable *hlo_share_common_subexpressions* is used. The value of this attribute determines whether common subexpressions are shared during compile, to reduce the cost of the design. Setting the attribute to *false* overrides the *hlo_share_common_subexpressions*. Set with *set_share_cse*.

structure

Determines if a design is to be structured during *compile*. If *true*, adds logic structure to a design by adding intermediate variables that are factored out of the design's equations. Set with *set_structure .*

structure_boolean

Enables the use of Boolean (non-algebraic) techniques during the structuring phase of optimization. This attribute is ignored if the *structure* attribute is *false* Set with *set_structure* .

structure_timing

Enables timing constraints to be considered during the structuring phase of optimization. This attribute is ignored if the *structure* attribute is *false*. Set with *set_structure* .

ungroup

Removes a level of hierarchy from the current design by exploding the contents of the specified cell in the current design. Set with *set_ungroup* .

wired_logic_disable

When *true*, disables creation of wired OR logic during *compile*. The default is *false*; if this attribute is not set, wired OR logic will be created if appropriate. Set with *set_wired_logic_disable*.

wire_load_model_mode

Determines which wire load model to use to compute wire capacitance, resistance, and area for nets in a hierarchical design that has different wire load models at different hierarchical levels. Allowed values are *top*, which indicates to use the wire load model at the top hierarchical level; *enclosed*, which indicates to use the wire load model on the smallest design that encloses a net completely; and *segmented*, which indicates to break the net into segments, one within each hierarchical level. In the *segmented* mode, each net segment is estimated using the wire load model on the design that encloses that segment. The *segmented* mode is not supported for wire load models on clusters. If a value is not specified for this attribute, *compile* searches for a default in the first library in the link path. If none is found, *top* is the default. Set with *set_wire_load*.

xnfout_use_blknames

When *true*, the Synopsys XNF writer writes BLKNM XNF parameters into the XNF netlist for your design when *write -f xnf* is invoked. The default is *false*. The BLKNM XNF parameters convey to the Xilinx place and route tools information, previously placed on the *db_design* by *replace_fpga*, that indicates which groupings of function generators are to be packed into CLB cells. Set with *set_attribute*.

Feedback

Library Cell Attributes

dont_touch

Identifies library cells to be excluded from optimization. Values are *true* (the default) or *false*. Library cells with the *dont_touch* attribute set to *true* are not modified or replaced during *compile*. Setting *dont_touch* on a hierarchical cell sets the attribute on all cells below it. Set with *set_dont_touch .*

dont_use

Disables the specified library cells so that they are not added to a design during *compile*. Set with *set_dont_use .*

formula

The attribute of the priority parameter for implementations in synthetic libraries. The formula should evaluate to an integer between 0 and 10. Set with *set_impl_priority*.

implementation

Specifies the implementation for the synthetic library cell instances to use. When compile is run, the implementation you specified is used if you set this attribute. The cells instances must be defined in the synthetic library for this attribute to work. Set with *set_implementation*.

no_sequential_degenerates

When *true*, disables mapping to versions of this latch or flip flop that have some input pins connected to 0 or to 1. Set with *set_attribute*. This attribute may also be set on the library itself, and that value will apply as the default for all registers in the library which don't have the attribute set individually.

preferred

Specifies the preferred library gate to use during technology translation when there are other gates with the same function in the target library. Set with the *set_prefer* command.

scan

When *true*, specifies that the instances of the library cell are always replaced by equivalent scan cells during *insert_dft*. When false, instances are not replaced. Set with *set_scan.*

scan_group

A user-defined string variable that allows you to specify to DFT Compiler a preferred scan equivalent for a non-scan storage element, when a library contains multiple scan equivalents. Typical values are *low*, *medium*, and *high*, for low, medium, and high drive strengths. However, you can define any string

variable, and it need not describe drive strength. The default behavior is for DFT Compiler to attempt to choose a scan element that best matches the electrical characteristics of the nonscan element; for a more detailed explanation, refer to the *DFT Compiler Scan Synthesis User Guide*. The matching of electrical characteristics works well with the standard CMOS delay model, but is not accurate with other delay models; *scan_group* provides a means for you to specify an appropriate scan equivalent. Normally, *scan_group* would be set by the ASIC vendor or library developer, but can also be set by you. Consult your ASIC vendor before attempting to set *scan_group* with *set_attribute*. For more information about *scan_group*, refer to the *DFT Compiler Scan Synthesis User Guide*.

set_id

Allows for the value for the implementations in synthetic libraries. Set with *set_impl_priorities*.

scan_element

Determines if specified designs are scan replaced by *insert_scan*. Set using *set_scan_element*.

scan_latch_transparent

When *true*, makes the specified library cells transparent in ATPG. For hierarchical cells, the effects apply hierarchically to level-sensitive leaf cells. The *set_scan_transparent* command sets the attribute; the *remove_attribute* command removes it.

sequential_bridging

When *true*, enables *Design Compiler* to take a multiplexed flip-flop and bridge (that is, connect) the output to the input to get a desired functionality. The default is *false*, so this attribute must be set in order to enable the functionality. Bridging is required for mapping in cases where there is no flip-flop with internal feedback in the target library but one is desired in the HDL. Set with *set_attribute*. This attribute may also be set on the library itself, and that value will apply as the default for all registers in the library which don't have the attribute set individually.

NOTE: Setting this attribute to *true* can result in an increase in run times and memory consumption for Design Compiler. The increased run times depend on the number of flip-flops in the target library or libraries for which this attribute has been set.

Pin Attributes

actual_max_net_capacitance

actual_min_net_capacitance

A floating point number that specified the total calculated capacitance of the net that is connected to the given pin. The attributes are defined only for pins of leaf cell. The value of these attributes is calculated upon request. These are "read-only" attributes and they cannot be set by the user.

disable_timing

Disables timing arcs. This has the same effect on timing as not having the arc in the library. Set with *set_disable_timing*.

max_slack

A floating point value representing the worst slack of *max_rise_slack* and *max_fall_slack*.

max_fall_slack

A floating point value representing the worst slack at a pin for falling maximum path delays. This attribute is valid for any pin that appears in a constrained path after timing has been updated.

max_rise_slack

A floating point value representing the worst slack at a pin for rising maximum path delays. This attribute is valid for any pin that appears in a constrained path after timing has been updated.

min_slack

A floating point value representing the worst slack of *min_rise_slack* and *min_fall_slack*.

min_fall_slack

A floating point value representing the worst slack at a pin for falling minimum path delays. This attribute is valid for any pin that appears in a constrained path after timing has been updated..

min_rise_slack

A floating point value representing the worst slack at a pin for rising minimum path delays. This attribute is valid for any pin that appears in a constrained path after timing has been updated.

max_fall_delay

A floating point value that specifies the maximum falling delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_max_delay*.

max_rise_delay

A floating point value that specifies the maximum rising delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_max_delay*.

min_fall_delay

A floating point value that specifies the minimum falling delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_min_delay*.

min_rise_delay

A floating point value that specifies the minimum rising delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_min_delay*.

hold_uncertainty

Specifies a negative uncertainty from the edges of the ideal clock waveform. Affects all sequential cells in the transitive fanout of this pin. Set with *set_clock_uncertainty -hold*.

observe_pin

Specifies the (internal) observe pin name of an LSI Logic scan macrocell (LSI CTV only). This attribute is used by the *write_test* command. Set with *set_attribute .*

pin_direction

Specifies the direction of a pin. Allowed values are *in*, *out*, *inout*, or *unknown*. This attribute is *read-only* and cannot be set by the user.

pin_properties

Lists valid EDIF property values to be attached to different versions of the output pin. The EDIF property values correspond to different output emitter-follower resistance values on the output pin. For details about the use of this attribute, refer to the *Library Compiler Reference Manual*, Chapter 6, "Defining Cells." Set with *set_attribute*.

setup_uncertainty

Specifies a positive uncertainty from the edges of the ideal clock waveform. Affects all sequential cells in the transitive fanout of this pin. Set with *set_clock_uncertainty -setup*.

set_pin

Specifies the (internal) set pin name of an LSI Logic scan macrocell (LSI CTV only). This attribute is used by the *write_test* command. Set with *set_attribute .*

signal_type

Used to indicate that a pin or port is of a special type, such as a *clocked_on_also* port in a master/slave clocking scheme, or a *test_scan_in* pin for scan-test circuitry. Set with *set_signal_type* .

static_probability

A floating point number that specifies the percentage of time that the signal is in the logic 1 state; this information is used by *report_power*. If this attribute is not set, *report_power* will use the default value of 0.5, indicating that the signal is in the logic 1 state half the time. Set with *set_switching_activity*.

test_assume

A string that represents a constant logic value to be assumed for specified pins throughout test design rule checking by *check_test*. "1", "one", or "ONE" specifies a constant value of logic one; "0", "zero", or "ZERO" specifies a constant value of logic zero. Use *report_test -assertions* for a report on objects that have the *test_assume* attribute set. Set with *set_test_assume* .

test_initial

A string that represents an initial logic value to be assumed for specified pins at the start of test design rule checking and fault simulation by *check_test*. "1", "one", or "ONE" specifies an initial value of logic one; "0", "zero", or "ZERO" specifies an initial value of logic zero. Use *report_test -assertions* for a report on objects that have the *test_initial* attribute set. Set with *set_test_initial*.

test_isolate

Indicates that the specified sequential cells, pins, or ports are to be logically isolated and considered untestable during test design rule checking by *check_test*. When this attribute is set on a cell, it is also placed on all pins of that cell. Do not set this attribute on a hierarchical cell. Use *report_test -assertions* for a report on isolated objects. Set with *set_test_isolate*.

Note: Setting this attribute suppresses the warning messages associated with the isolated objects.

test_routing_position

Specifies the preferred routing order of the scan-test signals of the identified cells. Set with *set_test_routing_order* .

toggle_rate

A positive floating point number that specifies the toggle rate; that is, the number of zero-to-one and one-to-zero transitions within a library time unit period. This information is used by *report_power*; if this attribute is not set, *report_power* will use the default value of 2*(static_probability)(1 - static_probability). The

default will be scaled by any associated clock signal (if one is available). Set with *set_switching_activity*.

true_delay_case_analysis

Specifies a value to set all or part of an input vector for *report_timing -true* and *report_timing -justify*. Allowed values are *0*, *1*, *r* (rise, X to 1), and *f* (fall, X to 0). Set with the *set_true_delay_case_analysis* command.

Port Attributes

actual_max_net_capacitance

actual_min_net_capacitance

A floating point number that specified the total calculated capacitance of the net connected to the given port. The value of these attributes is calculated upon request. These are "read-only" attributes and they cannot be set by the user.

connection_class

A string that specifies the connection class label to be attached to a port or to a list of ports. *compile*, *insert_pads*, and *insert_dft* will connect only those loads and drivers that have the same connection class label. The labels must match those in the library of components for the design, and must be separated by a space. The labels *universal* and *default* are reserved; *universal* indicates that the port can connect with any other load or driver, and *default* is assigned to any ports that do not have a connection class already assigned. Set with *set_connection_class*.

dont_touch_network

When a design is optimized, *compile* assigns *dont_touch* attributes to all cells and nets in the transitive fanout of *dont_touch_network* clock objects. The *dont_touch* assignment stops at the boundary of storage elements. An element is recognized as storage only if it has setup or hold constraints. Set with *set_dont_touch_network .*

driven_by_dont_care

Specifies that input port are driven by dont_care. Compile uses this information to create smaller designs. After optimization, the port connected to dont_care does not drive anything inside the optimized design. Set with *set_logic_dc*.

driven_by_logic_one

Specifies that input ports are driven by logic one. *compile* uses this information to create smaller designs. After optimization, a port connected to logic one usually does not drive anything inside the optimized design. Set with *set_logic_one .*

driven_by_logic_zero

Specifies that input ports are driven by logic zero. *compile* uses this information to create smaller designs. After optimization, a port connected to logic zero usually does not drive anything inside the optimized design. Set with *set_logic_zero .*

driving_cell_dont_scale

When *true*, indicates not to scale the transition time on the port using the driving cell. Otherwise the transition time will be scaled by operating condition factors. Set with *set_driving_cell*.

driving_cell_fall

A string that names a library cell from which to copy fall drive capability to be used in fall transition calculation for the port. Set with *set_driving_cell*.

driving_cell_from_pin_fall

A string that names the driving_cell_fall input pin to be used to find timing arc fall drive capability. Set with *set_driving_cell*.

driving_cell_from_pin_rise

A string that names the driving_cell_rise input pin to be used to find timing arc rise drive capability. Set with *set_driving_cell*.

driving_cell_library_fall

A string that names the library in which to find the *driving_cell_fall*. Set with *set_driving_cell*.

driving_cell_library_rise

A string that names the library in which to find the *driving_cell_rise*. Set with *set_driving_cell*.

driving_cell_multiply_by

A floating point value by which to multiply the transition time of the port marked with this attribute. Set with *set_driving_cell*.

driving_cell_pin_fall

A string that names the driving_cell_fall output pin to be used to find timing arc fall drive capability. Set with *set_driving_cell*.

driving_cell_pin_rise

A string that names the driving_cell_rise output pin to be used to find timing arc rise drive capability. Set with *set_driving_cell*.

driving_cell_rise

A string that names a library cell from which to copy rise drive capability to be used in rise transition calculation for the port. Set with *set_driving_cell*.

fall_drive

Specifies the drive value of high to low transition on input or inout ports. Set with *set_drive.*

fanout_load

Specifies the fanout load on output ports. Set with *set_fanout_load .*

load

Specifies the load value on ports. The total load on a net is the sum of all the loads on pins, ports, and wires associated with that net. Set with *set_load.*

max_capacitance

A floating point number that sets the maximum capacitance value for input, output, or bidirectional ports, and/or designs. The units must be consistent with those of the logic library used during optimization. Set with *set_max_capacitance*.

max_fall_delay

A floating point value that specifies the maximum falling delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_max_delay*.

max_fanout

Specifies the maximum fanout load for the net connected to this port. *compile* ensures that the fanout load on this net is less than the specified value. Set with *set_max_fanout .*

max_rise_delay

A floating point value that specifies the maximum rising delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_max_delay*.

max_slack

A floating point value representing the worst slack of *max_rise_slack* and *max_fall_slack*.

max_fall_slack

A floating point value representing the worst slack at a pin for falling maximum path delays. This attribute is valid for any pin that appears in a constrained path after timing has been updated.

max_rise_slack

A floating point value representing the worst slack at a pin for rising maximum path delays. This attribute is valid for any pin that appears in a constrained path after timing has been updated.

min_slack

A floating point value representing the worst slack of *min_rise_slack* and *min_fall_slack*.

min_fall_slack

A floating point value representing the worst slack at a pin for falling minimum path delays. This attribute is valid for any pin that appears in a constrained path after timing has been updated.

min_rise_slack

A floating point value representing the worst slack at a pin for rising minimum path delays. This attribute is valid for any pin that appears in a constrained path after timing has been updated.

max_time_borrow

A floating point number that establishes an upper limit for time borrowing; that is, it prevents the use of the entire pulse width for level-sensitive latches. Units are those used in the logic library. Set with *set_max_time_borrow*.

max_transition

Specifies the maximum transition time for the net connected to this port. *compile* ensures that value. Set with *set_max_transition .*

min_capacitance

A floating point number that sets the minimum capacitance value for input and/or bidirectional ports. The units must be consistent with those of the logic library used during optimization. Set with *set_min_capacitance*.

min_fall_delay

A floating point value that specifies the minimum falling delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_min_delay*.

min_rise_delay

A floating point value that specifies the minimum rising delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_min_delay*.

setup_uncertainty

Specifies a positive uncertainty from the edges of the ideal clock waveform. Affects all sequential cells in the transitive fanout of this port. Set with *set_clock_uncertainty -setup*.

hold_uncertainty

Specifies a negative uncertainty from the edges of the ideal clock waveform. Affects all sequential cells in the transitive fanout of this port. Set with *set_clock_uncertainty -hold*.

model_drive

A non-negative floating point number that specifies the estimated drive value on ports in terms of standard drives of the current logic library. Set with *set_model_drive*.

model_load

A non-negative floating point number that specifies the estimated load value on ports in terms of standard loads of the current logic library. Set with *set_model_load*.

op_used_in_normal_op

Specifies that a scan-out port is also used in normal operation (system mode). This attribute is used by the *insert_dft* command. Set with *set_attribute*.

Read-Only Attributes

design_type

Indicates the current state of the design and has the value *fsm* (finite state machine), *pla* (programmable logic array), *equation* (Boolean logic), or *netlist* (gates). This attribute cannot be set by the user.

is_black_box

true if the reference is not yet linked to a design or is linked to a design that doesn't have a functionality. This attribute cannot be set by the user.

is_combinational

true if all cells of a design and all designs in its hierarchy are combinational. A cell is combinational if it is non-sequential or non-tristate and all of its outputs compute a combinational logic function. The *report_lib* command will report such a cell as not a black-box. This attribute is read-only and cannot be set by the user.

Feedback

is_dw_subblock

true if the object (a cell, a reference, or a design) is a DW subblock that was automatically elaborated. This attribute is "read-only" and cannot be set by the user.

is_hierarchical

true if any of the cells of a design are not leaf cells (for example, not from a logic library). This attribute cannot be set by the user.

is_mapped

true if all the non-hierarchical cells of a design are mapped to cells in a logic library. This attribute cannot be set by the user.

is_sequential

true if any cells of a design or designs in its hierarchy are sequential. A cell is sequential if it is not combinational. This attribute cannot be set by the user.

is_synlib_module

true if the object (a cell, a reference, or a design) refers to an unmapped module reference or if the object is (or refers to) a design that was automatically elaborated from a synlib module or a synlib operator. This attribute is "read-only" and cannot be set by the user.

NOTE: synlib modules that are manually elaborated will not have this attribute.

is_synlib_operator

true if the object (a cell or a reference) is a synthetic library operator reference. This attribute is "read-only" and cannot be set by the user.

is_test_circuitry

Set by *insert_dft* on the scan cells and nets added to a design during the addition of test circuitry. This attribute cannot be set by the user.

is_unmapped

true if any of the cells are not linked to a design or mapped to a logic library. This attribute cannot be set by the user.

direction

Returns the direction of a pin or port. The value can be *in*, *out*, *inout*, or *unknown*. For backward compatibility, the attribute also supports an integer value, *1* for *in*, *2* for *out*, or *3* for *inout*. This attribute cannot be set by the user.

pin_direction

Returns the direction of a pin. The value can be *in*, *out*, *inout*, or *unknown*. This attribute cannot be set by the user. You can also use the *direction* attribute to get the direction of the pin.

port_direction

Returns the direction of a port. The value can be *in*, *out*, *inout*, or *unknown*. This attribute cannot be set by the user. You can also use the *direction* attribute to get the direction of the port.

ref_name

The reference name of a cell. This attribute cannot be set by the user.

Reference Attributes

dont_touch

Specifies that designs linked to a reference with this attribute are excluded from optimization. Values are *true* (the default) or *false*. Designs linked to a reference with the *dont_touch* attribute set to *true* are not modified or replaced during *compile*. Set with *set_dont_touch .*

is_black_box

true if the reference is not yet linked to a design or is linked to a design that doesn't have a functionality. This attribute is read-only and cannot be set by the user.

is_combinational

true if all the cells of the referenced design are combinational. A cell is combinational if it is non-sequential or non-tristate and all of its outputs compute a combinational logic function. The *report_lib* command will report such a cell as not a black-box. This attribute is read-only and cannot be set by the user.

is_dw_subblock

true if the object (a cell, a reference, or a design) is a DW subblock that was automatically elaborated. This attribute is read-only and cannot be set by the user.

NOTE: DW subblocks that are manually elaborated will not have this attribute.

is_hierarchical

true if the referenced design is not a leaf cell (for example, not in a logic library). This attribute is read-only and cannot be set by the user.

is_mapped

true if the reference is linked to a design, and all the non-hierarchical cells of the referenced design are mapped to cells in a logic library. This attribute is read-only and cannot be set by the user.

is_sequential

true if all the cells of the referenced design are sequential. A cell is sequential if it is not combinational (if any of its outputs depend on previous inputs). This attribute is read-only and cannot be set by the user.

is_synlib_module

true if the object (a cell, a reference, or a design) refers to an unmapped module reference or if the object is (or refers to) a design that was automatically elaborated from a synlib module or a synlib operator. This attribute is read-only and cannot be set by the user.

NOTE: synlib modules that are manually elaborated will not have this attribute.

is_synlib_operator

true if the object (a cell or a reference) is a synthetic library operator reference. This attribute is read-only and cannot be set by the user.

is_unmapped

true if any of the non-hierarchical cells of the referenced design are not mapped to cells in a logic library, or if the reference is not yet linked to a design. This attribute is read-only and cannot be set by the user.

scan

When *true*, specifies that cells of the referenced design are always replaced by equivalent scan cells during *insert_dft*. When false, cells of the design are not replaced. Set with *set_scan .*

scan_chain

Includes the specified cells of the referenced design in the scan-chain whose index is the value of this attribute. Set with the *set_scan_chain* command.

scan_element

Determines if specified designs are scan replaced by *insert_scan*. Set using *set_scan_element*.

scan_latch_transparent

When *true*, makes the specified references transparent in ATPG. For hierarchical cells, the effects apply hierarchically to level-sensitive leaf cells.

Feedback

The specified library cell cannot be overwritten. Set with *set_scan_transparent*; remove with *remove_attribute*.

ungroup

Specifies that all designs linked to a reference with this attribute are ungrouped (levels of hierarchy represented by these design cells are removed) during *compile*. Set with *set_ungroup .*

See Also

- get_attribute
- remove_attribute
- set_attribute
- power_attributes

---

auto_insert_level_shifters

Controls automatic level shifter insertion.

Data Types

Boolean

Default true

Group

mv

Description

By default, this variable is *true* and enables the level-shifter insertion engine. With this variable set to *true*, commands such as *compile* and *insert_dft* are able to automatically insert level shifters where needed.

Setting this variable to *false* disables the level-shifter insertion engine, and automatic level shifter insertion does not happen during *compile* or *insert_dft* regardless of any user constraint.

Note that level-shifter insertion performed by the command *insert_mv_cells* is not affected by this variable.

Feedback

See Also

- compile
- insert_dft
- insert_mv_cells

---

auto_insert_level_shifters_on_clocks

Directs automatic level-shifter insertion to insert level shifters on specified clocks.

Data Types

string

Default ""

Group

none

Description

This variable can be set to "all" or a list of clock names (delimited by spaces or commas). When set to "all", automatic level-shifter insertion inserts level shifters on all clock nets that need level shifters.

When this variable is set to a list of clock names, automatic level-shifter insertion inserts level shifters on the net of these clocks, if needed.

By default, automatic level-shifter insertion does not insert level shifters on the nets driven by the clocks.

Note that if a net is ideal, automatic level-shifter insertion won't insert any level shifter on it, unless the variable *mv_insert_level_shifters_on_ideal_nets* is set to "all".

See Also

- auto_insert_level_shifters
- create_clock
- mv_insert_level_shifters_on_ideal_nets
- set_ideal_network

---

auto_link_disable

Specifies whether to disable the code to perform an auto_link during any command.

Feedback

Data Types

Boolean

Default false

Description

When this variable is set to true, the code to perform an auto_link during any command is disabled, resulting in faster command processing. This increase in speed is important in back-annotation commands such as *set_load*, *set_resistance*, and *set_annotated_delay*, where numerous commands are executed in sequence. Disabling the auto_link code can significantly improve the speed with which commands are executed.

Once the sequence of time-critical commands is completed, reset the variable value to false to revert the tool back to its normal mode of operation.

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

See Also

- set_annotated_delay
- set_load
- set_resistance

---

auto_link_options

Specifies the *link* command options to be used when *link* is invoked automatically.

Data Types

string

Default -all

Description

This variable specifies the *link* command options to be used when *link* is invoked automatically. The default value is -all. To find the available options, refer to the *link* command man page.

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

See Also

- link

Feedback

auto_ungroup_preserve_constraints

Controls whether the timing constraints on the hierarchy are preserved when the hierarchy is ungrouped during the process of optimization.

Data Types

Boolean

Default true

Group

timing

Description

This variable enables the ungrouping of hierarchies with timing constraints and preserves the timing constraints when the hierarchies are ungrouped during the process of optimization. By default, the *auto_ungroup_preserve_constraints* variable is set to true.

The constraints are preserved when executing the following commands:

```
prompt> compile -ungroup_all

prompt> set_ungroup
prompt> compile

prompt> compile -auto_ungroup area | delay
```

The constraints on DesignWare library hierarchical cells are also lost when the cells are ungrouped during optimization.

Set the *auto_ungroup_preserve_constraints* variable to false before compiling to avoid the ungrouping of hierarchies with timing constraints.

Use the *printvar auto_ungroup_preserve_constraints* command to determine the current value of this variable.

See Also

- compile
- list
- set_ungroup

auto_wire_load_selection

Controls automatic selection of wire load model.

Feedback

Data Types

string

Default true in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

When area_locked, the automatic wire load selection uses the initial area to do the first selection of the wire load model and then adjusts the wire load model down if the area drops. When area_reselect, the automatic wire load selection during reporting and at various points in compile updates the wire load model to the current area of the design. When false, the automatic wire load selection is off. For backwards compatibility, we also support the value true. True is the same as area_locked.

The automatic selection of the wire load model is used to estimate net capacitances and resistances from the net fanout. The wire load models are described in the technology library. With the automatic selection of the wire load model, if the wire load mode is *segmented* or *enclosed*, the wire load model will be chosen based on the area of the design containing the net either partially (for *segmented*) or fully (for *enclosed*). If the wire load mode is *top*, the wire load model will be chosen based on the area of the top level design for all nets in the design hierarchy.

When a design's wire load model is selected manually by the user (with the command *set_wire_load*), no wire load is selected automatically for that design.

To determine the current value of this variable use *printvar auto_wire_load_selection*. For a list of all *compile* variables and their current values, use the *print_variable_group compile* command.

See Also

- set_wire_load
- report_design

b

banking_enable_attribute_spreading

Controls the spreading of attributes during the execution of the *create_register_bank* and *split_register_bank* commands.

Feedback

Data Types

Boolean

Default true

Description

By default the tool spreads the values of the attributes present in the original cells so that the affected netlist entities are up-to-date immediately after the command execution. This may have a runtime impact in some corner case situations, and so this variable provides the user a way to disable that spreading so that it happens later in the flow after all the multi-bit transformations are finished.

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

```
prompt> printvar banking_enable_attribute_spreading
```

See Also

- identify_register_banks
- create_register_bank
- split_register_bank

---

banking_enable_concatenate_name

Controls the multibit cell name used for the *create_register_bank* commands in the output file generated by the *identify_register_banks* command.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

true in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Starting with the J-2014.09-SP3 release, the *banking_enable_concatenate_name* variable controls the name of the multibit registers created by the *create_register_bank* command in the output file generated by the *identify_register_banks* command. By default, when the single-bit registers are grouped together, the tool uses the following naming style for multibit cells: The name of the original single-bit registers concatenated with an underscore (_), such as reg_1_reg_0.

Feedback

For example, if the tool groups two single-bit registers, a_reg[0] and a_reg[1], to one multibit cell, it generates the following in the output file:

create_register_bank -name a_reg[0]_a_reg[1] {a_reg[0] a_reg[1]} -lib_cell MREG2

To revert to the naming style used in previous releases, set the *banking_enable_concatenate_name* variable to *false*. When the variable is *false*, the name of each register bank is created using the *prefixN1_N2* format, where N1 and N2 are integers. The N1 and N2 integers are used so the name is unique in the output file. The default prefix is group.

In the previous naming style, if the tool groups two single-bit registers, a_reg[0] and a_reg[1], to one multibit cell, the following command is generated in the output file:

create_register_bank -name group0_0 {a_reg[0] a_reg[1]} -lib_cell MREG2

You can change the name prefix in either naming style by specifying the *-name_prefix* option of the *identify_register_banks* command.

For example, if you specify the MBIT prefix with the *-name_prefix* option using the default naming style, the output file generated during placement includes the following *create_register_bank* command:

create_register_bank -name MBIT_a_reg[0]_a_reg[1] {a_reg[0] a_reg[1]} -lib_cell MREG2

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

```
prompt> printvar banking_enable_concatenate_name
```

See Also

- identify_register_banks

---

bind_unused_hierarchical_pins

Specifies if unused hierarchical pins (no driver and no load) should be connected to constant tie-off cells during compile and insert_dft.

Data Types

Boolean

Default true

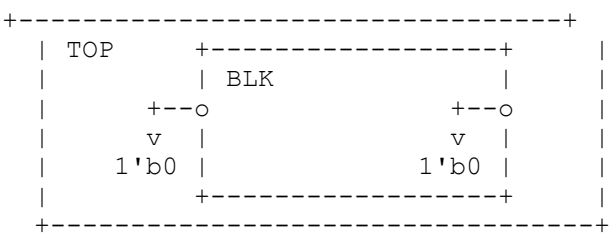
Description

This variable controls whether unused hierarchical pins should connect to constant tie-off cells during compile and insert_dft.

Feedback

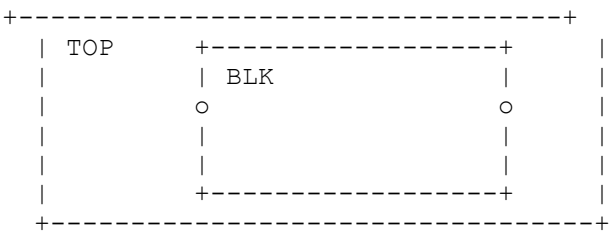
An unused hierarchical pin is a pin of a hierarchical cell that has no drivers and no loads associated with it. This means that there are no leaf cells and no ports connected to the hierarchical pin. Nets and connections to other hierarchical pins are not accounted.

When you set this variable to true (the default), the tool ties unused hierarchical pins to logic 0 during compile and insert_dft, as shown in the following diagram:



For unused input pins, the constant is driven in the parent hierarchy. For unused output pins, the constant is driven from inside the hierarchical block.

When you set this variable to *false* before linking the design, the tool leaves unused hierarchical pins undriven, as shown in the following diagram:



The variable affects only the *compile*, *compile_ultra*, and *insert_dft* commands. This variable does not affect design read, link, or design write commands.

Note that when you perform boundary optimization, constant propagation can result in unused hierarchical pins in the compiled design that were driven in the RTL. Boundary optimization is on by default when you use the *compile_ultra* command.

To determine the current value of this variable, use the *printvar bind_unused_hierarchical_pins* variable. For a list of all compile variables and their current values, use the *print_variable_group compile* command.

See Also

- compile
- insert_dft

Feedback

bit_blasted_bus_linking_naming_styles

Specifies the bus pin naming styles to be matched.

Data Types

String

Default %s[%d] %s(%d) %s_%d_

Description

When the *enable_bit_blasted_bus_linking* variable is set to true, the *bit_blasted_bus_linking_naming_styles* variable allows the linker to recognize bit-blasted buses by pin names during the link process. If the pin names follow a specific pattern, DC Explorer infers a bus when encountering the individual blasted bits.

Typically, the RTL pin names and the logic library pin names should match. Signals are defined the same way in both the RTL and the library. They are defined as buses or a bus is defined by its individual wires. Occasionally, mismatches occur during design development; for example, when you transfer the design from one technology to another. To fix the mismatches, you can set the *enable_bit_blasted_bus_linking* variable to true before the design is read.

Setting the *enable_bit_blasted_bus_linking* variable to true allows the linker to match buses and bit-blasted pin names according to the patterns specified by the *bit_blasted_bus_linking_naming_styles* variable. The default for the *enable_bit_blasted_bus_linking* variable is false.

For example,

```
prompt> set bit_blasted_bus_linking_naming_styles {%s[%d] %s_%d}
```

This setting permits the following U1 reference to link

```
input [1:0] in;
my_macro U1( .S(in),...
```

even if my_macro is defined as follows:

```
my_macro
input S_1;
input S_0;
\...
```

For multiple-dimension buses, the style can be explicitly defined or in certain cases implied. For example, the %s[%d] setting matches

A[0], A[0][0], or A[0][0][0], and so on.

Feedback

However, to match the A_0__0__0_ pin name, you need to use the following setting:

%s_%d__%d__%d_

To determine the current setting of this variable, use the *printvar bit_blasted_bus_linking_naming_styles* command. For a list of all system variables and their current values, use *print_variable_group system*.

See Also

- enable_bit_blasted_bus_linking

bit_blasted_bus_linking_order

Specifies the pin order for bus inference of bit blasted bus pins.

Data Types

String

Default reference

Description

When the *enable_bit_blasted_bus_linking* variable is set to true, the linker can recognize bit-blasted buses by pin names during the link process. If the pin names follow a specific pattern, DC Explorer infers a bus when encountering the individual blasted bits. The *bit_blasted_bus_linking_order* variable allows the user to specify the bus order.

The allowed values are {reference|ascending|descending}. when the variable value is set to 'ascending' or 'descending', a reconstructed 32 bit bus A would become, respectively, A[0:31] and A[31:0]. If the variable value is 'reference' the reconstructed bus pin order will follow the order found on the bit blasted reference. The variable defines the global behavior for all linked designs in the session.

To determine the current setting of this variable, use the *printvar bit_blasted_bus_linking_order* command. For a list of all system variables and their current values, use *print_variable_group system*.

See Also

- enable_bit_blasted_bus_linking

block_abstraction_compute_area_as_macro

Controls the calculation of the area of cells in a block abstraction with the *report_area* and *report_qor* commands.

Feedback

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

By default, this variable is set to *false*. The area of cells in a block abstraction is calculated and reported in the regular cell portions by the *report_area* and *report_qor* commands. If this variable is set to *true*, the area of cells is calculated and reported in the macro portion.

See Also

- report_area
- report_qor

---

bsd_max_in_switching_limit

Specifies the maximum number of design inputs that may switch simultaneously while generating input Design Compiler parametric tests using the *create_bsd_patterns* command.

Data Types

integer

Default 60000 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

Switching all inputs and outputs simultaneously can cause large currents. It is advisable to switch as few I/Os as possible while performing Design Compiler parametric tests. The *bsd_max_in_switching_limit* variable allows you to specify the maximum number of inputs that may switch simultaneously while generating VIL/VIH tests using the *create_bsd_patterns* command.

To determine the current value of this variable, use the *printvar bsd_max_in_switching_limit* command. For a list of BSD variables and their current values, use the *print_variable_group bsd* command.

Feedback

See Also

- create_bsd_patterns

---

bsd_max_out_switching_limit

Specifies the maximum number of design outputs that may switch simultaneously while generating output Design Compiler parametric tests using the *create_bsd_patterns* command.

Data Types

integer

Default 60000 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

Switching all inputs and outputs simultaneously can cause large currents. It is advisable to switch as few I/Os as possible while performing Design Compiler parametric tests. The *bsd_max_out_switching_limit* variable allows you to specify the maximum number of outputs that may switch simultaneously while generating VOL/VOH tests using the *create_bsd_patterns* command.

To determine the current value of this variable, use the *printvar bsd_max_out_switching_limit* command. For a list of BSD variables and their current values, use the *print_variable_group bsd* command.

See Also

- create_bsd_patterns

---

bsd_physical_effort

Specifies the effort used for BSD Compiler integration features.

Data Types

string

Default medium

Description

Specifies the effort used for BSD Compiler integration features.

Feedback

Valid values for this variable are low, medium, high, and ultra. The default value is medium.

To determine the current value of this variable, use the *printvar bsd_physical_effort* command. For a list of BSD variables and their current values, use the *print_variable_group bsd* command.

---

budget_generate_critical_range

Enables automatic generation of *set_critical_range* commands by *dc_allocate_budgets* for multiply-instantiated subdesigns.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

When set to true, this variable causes the *dc_allocate_budgets* command to automatically generate a *set_critical_range* command for any budgeted cells that have multiply-instantiated subdesigns. The value of *critical_range* is set to 10% of the shortest clock period in the design. If the top-level design already has a *critical_range* attribute, then that original value is used instead.

The purpose of this is to improve QoR, since any multiple instances must be dont_touched after compile in a bottom-up compile flow. Using *set_critical_range* causes any near-critical paths in these blocks to be optimized before the block is dont_touched.

To see the current value of this variable, enter the following command:

```
prompt> printvar budget_generate_critical_range
```

See Also

- dc_allocate_budgets

---

budget_map_clock_gating_cells

Maps integrated clock gating cells into target library cells during RTL budgeting.

Data Types

Boolean

Feedback

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

When set to true, this variable causes the *dc_allocate_budgets -mode rtl* command to map any integrated clock gating cells into target library cells before calculating the budget. Normally, any unmapped cells would remain unmapped during budgeting when *-mode rtl* is used.

The purpose of this is to prevent incorrect *-level_sensitive* delay constraints from appearing in the budget constraint files. These may otherwise appear because unmapped integrated clock gating cells can include transparent latch elements.

To see the current value of this variable, use the following command:

```
prompt> printvar budget_map_clock_gating_cells
```

See Also

- dc_allocate_budgets

---

bus_inference_descending_sort

Specifies that the members of that port bus are to be sorted in descending order rather than in ascending order.

Data Types

Boolean

Default true

Description

Affects the *read* command except for the db, Verilog and VHDL formats. This variable is primarily used when reading in designs in the LSI/NDL format. That particular format does not support representation of busses, but, if port names follow a specific pattern (as described in the variable *bus_inference_style*), the individual bits can be "inferred" into a port bus. When true (the default value), This variable specifies that the members of that port bus are to be sorted in descending order rather than in ascending order.

For example, with the variable *bus_inference_style* set to "%s[%d]", the ports "A[1]", "A[2]", "A[3]", and "A[4]" will be "inferred" into a port bus named "A". If this variable is true, the port bus "A" will have an index from 4 to 1; if this variable is false, the port bus "A" will have an index from 1 to 4.

Feedback

With the variable *bus_inference_style* set to "#%d%%%s", the ports "#8%cb", "#9%cb", "#10%cb", and "#11%cb" will be "inferred" into a port bus named "cb". If this variable is true, the port bus "cb" will have an index from 11 to 8; if this variable is false, the port bus "cb" will have an index from 8 to 11.

To determine the current value of this variable, type *printvar bus_inference_descending_sort*. For a list of all *edif* or *io* variables and their current values, type *print_variable_group edif* or *print_variable_group io*.

See Also

- create_bus
- remove_bus
- report_bus
- bus_inference_style
- bus_minus_style
- bus_naming_style
- bus_range_separator_style

---

bus_inference_style

Specifies the pattern used to infer individual bits into a port bus.

Data Types

string

Default ""

Description

This variable specifies the pattern used to infer individual bits into a port bus. The variable affects the *read* command except for the .db and VHDL formats. This variable also affects the VHDL *write* commands. The variable is used primarily when reading in designs in the LSI/NDL format. The LSI/NDL format does not support representation of buses. But if port names follow a specific pattern (as described by this variable), the individual bits can be inferred into a port bus. If you specify an invalid value, no port buses are inferred.

When running in Tcl mode, the *bus_inference_style* value must be specified within curly brackets ({}). For example:

```
set bus_inference_style {%s[%d]}
```

Feedback

This variable must contain 1 %s (percent s) and %d (percent d) character sequence. Additional characters can be used with these symbols. To use a percent sign in a name, 2 percent signs are needed in the variable string (%%).

In naming a port bus, the port name is substituted for %s, and the port number replaces %d. A single percent sign is substituted for %%.

For example, with this variable set to "%s[%d]", the ports "A[1]", "A[2]", "A[3]", and "A[4]" will be inferred into a port bus named A either with an index from 1 to 4, or with an index from 4 to 1 (see *bus_inference_descending_sort*).

With this variable set to "#%d%%%s", the ports "#8%cb", "#9%cb", "#10%cb", and "#11%cb" will be inferred into a port bus named cb either with an index from 8 to 11, or with an index from 11 to 8 (see *bus_inference_descending_sort*).

To determine the current value of this variable, use the *printvar bus_inference_style* command. For a list of all *edif* or *io* variables and their current values, use the *print_variable_group edif* or *print_variable_group io* command.

See Also

- create_bus
- remove_bus
- report_bus
- bus_inference_descending_sort
- bus_minus_style
- bus_naming_style
- bus_range_separator_style

---

bus_minus_style

Controls the naming of individual members of bit-blasted port, instance, or net buses with negative indices.

Data Types

string

Default -%d

Feedback

Description

This variable controls the naming of individual members of bit-blasted port, instance, or net buses with negative indices. The variable affects the *read* command with the *vhdl* format option.

To determine the current value of this variable, use the *printvar bus_minus_style* command. For a list of all HDL variables and their current values, use the *print_variable_group hdl* command.

See Also

- create_bus
- remove_bus
- report_bus
- bus_inference_descending_sort
- bus_inference_style
- bus_naming_style
- bus_range_separator_style

---

bus_multiple_name_separator_style

Determines the separator used to name of a multibit cell that implements bits whose original base_names differ.

Data Types

string

Default ,,

Description

This variable affects the naming of multibit cells during multibit mapping. The variable is used to name a multibit cell whose original single-bit cells had different base names. The default value is a double *bus_multiple_separator_style* pattern, and is used if an invalid value is specified (or no value at all).

The *bus_multiple_separator_style* variable is used to separate two ranges of bits sharing the same base names, while *bus_multiple_name_separator_style* is used to separate the different base names.

The two variables are used in conjunction with the *bus_naming_style* and the *bus_range_separator_style* variable to generate names for multibit cells.

Feedback

In the following examples assume that *bus_range_separator_style* is set to ":", *bus_multiple_name_separator_style* is set to "_MB_" and *bus_multiple_separator_style* is set to ",".

If cells with the names q_1[0], q_1[1], q_2[2], q_2[5], q_2[6], and q_2[7] are packed into a 6-bit wide cell, the name given to the new cell is q_1[0:1]_MB_q_2[2,5:7].

If cells q[0], q[2], q[4], and q[6] are packed into a 4-bit wide cell, the name given to the new cell is q[0,2,4,6].

To determine the current value of this variable, use the following command: *printvar bus_multiple_name_separator_style*.

For a list of all *multibit* variables and their current values, use: *print_variable_group multibit*

See Also

- bus_multiple_separator_style
- bus_range_separator_style

---

bus_multiple_separator_style

Determines the name of a multibit cell that implements bits that do not form a range.

Data Types

string

Default ,

Description

This variable affects the naming of multibit cells during multibit mapping. The variable is used to name a multibit cell that implements bits that do not form a range. The default is used if an invalid value is specified.

The *bus_range_separator_style* variable is used to separate the start and end bit positions of a range, while *bus_multiple_separator_style* is used to separate two ranges. The two variables are used in conjunction with the *bus_naming_style* variable to generate names for multibit cells.

Assume that *bus_range_separator_style* is set to ":", *bus_multiple_separator_style* is set to ",", and *bus_naming_style* is set to "%s[%d] in the following examples.

For example, if cells with the names q[0], q[1], q[2], q[5], q[6], and q[7] are packed into a 6-bit wide cell, the name given to the new cell is q[0:2,5:7]. If cells q[0], q[2], q[4], and q[6] are packed into a 4-bit wide cell, the name given to the new cell is q[0,2,4,6].

To determine the current value of this variable, use the *printvar bus_range_separator_style* command. For a list of all *multibit* variables and their current values, use the *print_variable_group multibit* command.

See Also

- bus_range_separator_style

---

bus_naming_style

Specifies the style to use in naming an individual port member, net member, or cell instance member of an EDIF array or of a Verilog or VHDL vector.

Data Types

string

Default %s[%d]

Description

This variable affects the *read* command with the EDIF, Verilog, or VHDL format option, the *write* command with the EDIF format option, and the *create_schematic* command with the busing option.

When reading buses, this variable specifies the style to use in naming an individual port member, net member, or cell instance member of an EDIF array or of a Verilog or VHDL vector.

When running in TCL mode the *bus_naming_style* needs to be enclosed by curly brackets. For example:

```
set bus_naming_style {%s[%d]}
```

When writing buses, this variable used with the *bus_range_separator_style* variable specifies the style to use in naming a port array or net array in the EDIF file. If you specify an invalid value, the array is given the name of the bus. When writing schematic nets, this variable used with the *bus_range_separator_style* variable specifies the style to use in naming a net connected to the "wire" end of a ripper in the EDIF file. If you specify an invalid value, the net is given the name of the original net.

When creating schematics, this variable used with the variable *bus_range_separator_style* variable specifies the style to use in naming a ripper, bused port, bused net, or net connected to the "wire" end of a ripper. If you specify an invalid value, the default value is used.

This variable must contain only 1 %s (percent s) and 1 %d (percent d) character sequence. To use the % (percent sign) in the name, use 2 of them in the variable string (%%). Therefore, the only characters that can follow a percent sign are %, s, or d.

When reading buses, in naming members, the name of the array is substituted for %s, and the number of the member is substituted for %d. One percent sign is substituted for %%.

For example, if this variable is set to "%s[%d]", then the first member of the 4-bit array "A", going from 0 to 3, is named:

```
A[0]
```

If this variable is set to "%s_%%%d.X", then the first member of the 8-bit array "xy", going from 9 to 16, is named:

```
xy_%9.X
```

See the *bus_dimension_separator_style* man page for a description of how it is used in conjunction with this variable for specifying the names of the members of multidimensional arrays in the EDIF format or multidimensional vectors in the VHDL format.

See the *bus_minus_style* man page for a description of how to specify the names of vectors with negative indices in the VHDL format.

When creating schematics or when writing buses, in naming a bused port or bused net or a port array or net array, the original bus or array name is substituted for %s. The start and end bits of the bus or array separated by the value of the variable *bus_range_separator_style* are substituted for %d. One percent sign is substituted for each %%.

For example, if this variable is set to "%s[%d:%d]", then the 4-bit bus or array "A", going from 0 to 3, is named:

```
A[0:3]
```

If this variable is set to "%%%s[%d][%d]", then the 8-bit bus or array "B", going from -4 to 3, is named:

```
%B[-4][3]
```

See the *edifout_multidimension_arrays* man page for a description of how it is used in conjunction with this variable for specifying the names of multidimensional arrays.

See the *edifout_numerical_array_members* man page for a description of how it is used in conjunction with this variable for specifying the names of descending arrays.

When creating schematics or when writing schematic nets, in naming a ripper or net connected to the "wire" end of a ripper, the original net name is substituted for %s. If the net is a scalar (a single bit) net, the bit of the ripper is substituted for %d. If the net is a bused net, the start and end bits of the ripper separated by the value of the variable

Feedback

bus_range_separator_style are substituted for %d. One percent sign is substituted for each %%.

For example, if this variable is set to "%s[%d]", then the ripper or the net connected to the "wire" end of the ripper that is ripping off the first bit of the 4-bit net array "A" going from 0 to 3, is named:

```
A[0]
```

If this variable is set to "%s_%%%d.X", then the ripper or the net connected to the "wire" end of the ripper that is ripping off the first bit of the 8-bit net array "xy" going from 9 to 16, is named:

```
xy_%9.X
```

If this variable is set to "%s[%d]" and the *bus_range_separator_style* variable is set to ":", then the net connected to the "wire" end of the ripper that is ripping off the third through fifth bits of the 8-bit net array "xy" going from 9 to 16, is named:

```
xy[11:13]
```

If this variable is set to "%s_%%%d.X" and the *bus_range_separator_style* variable is set to "..", then the ripper or the net connected to the "wire" end of the ripper that is ripping off the third through fourth bits of the 4-bit net array "A" going from 0 to 3, is named:

```
A_%2..3.X
```

To determine the current value of this variable, use the *printvar bus_naming_style* command. For a list of all EDIF, HDL, or schematic variables and their current values, use the *print_variable_group edif*, *print_variable_group hdl*, or *print_variable_group schematic* command.

See Also

- create_bus
- remove_bus
- report_bus
- bus_inference_descending_sort
- bus_inference_style
- bus_minus_style
- bus_range_separator_style

Feedback

bus_range_separator_style

Specifies the style to use in naming a net connected to the "wire" end of a ripper in the EDIF file.

Data Types

string

Default :

Description

This variable specifies the style to use in naming a net connected to the "wire" end of a ripper in the EDIF file. This variable affects the *write* command with the EDIF format option and the *create_schematic* command with the busing option.

When writing buses, this variable used with the *bus_naming_style* variable, specifies the style to use in naming a port array or net array in the EDIF file. When writing schematic nets, this variable used with the *bus_naming_style* variable, specifies the style to use in naming a net connected to the "wire" end of a ripper in the EDIF file.

When creating schematics, this variable used with the *bus_naming_style* variable, specifies the style to use in naming a ripper, bused port, bused net, or net connected to the "wire" end of a ripper. If you specify an invalid value, the default value is used.

See the *bus_naming_style* man page for a description of how this variable is used in conjunction with that variable.

To determine the current value of this variable, use the *printvar bus_range_separator_style* command. For a list of all EDIF or schematic variables and their current values, use the *print_variable_group edif* or *print_variable_group schematic* command.

See Also

- create_bus
- remove_bus
- report_bus
- bus_inference_descending_sort
- bus_inference_style
- bus_minus_style
- bus_naming_style

Feedback

c

case_analysis_log_file

Specifies the name of a log file generated during propagation of constant values, from case analysis or from nets tied to logic zero or logic one. Each scenario has its proprietary log file if multiple scenarios exist.

Data Types

string

Default ""

Description

This variable specifies the name of a log file generated during the propagation of constant values, from case analysis or from nets tied to logic zero or logic one. The log file contains the list of all ports and pins that propagate constants. The constant propagation algorithm is an iterative process that propagates constants through nets and cells starting from a list of constant pins. The algorithm finishes when no more constants can be propagated. The format of the log file follows the constant propagation algorithm.

In multicorner-multimode, you must specify the log file name in the definition of each scenario, or no log will be generated for that scenario. If you switch the active scenario, the log file used will be switched automatically. Log files for different scenarios can have the same name.

By default, this variable is set to an empty string, and no log file is generated during constant propagation.

To determine the current value of this variable, use the *printvar case_analysis_log_file* command. Note that if you switch the scenario in multicorner-multimode, the value of this variable remains at the value last set.

See Also

- remove_case_analysis
- report_case_analysis
- report_disable_timing
- set_case_analysis
- disable_case_analysis

Feedback

case_analysis_propagate_through_icg

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

Data Types

Boolean

Default false

Description

When this variable is set to false (the default), constants propagating throughout the design will 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 will propagate in the fanout of the cell.

When the value is true, constants propagated throughout the design will 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, for a latch_posedge ICG, when it is disabled, it will propagate a logic 0 in its fanout.

To activate logic propagation through all integrated clock gating cells, you must set the following:

set case_analysis_propagate_through_icg true

To determine the current value of this variable, use either the *printvar case_analysis_propagate_through_icg* or the *echo $case_analysis_propagate_through_icg* command.

See Also

- remove_case_analysis
- set_case_analysis

case_analysis_sequential_propagation

Determines whether case analysis is propagated across sequential cells.

Data Types

string

Default never

Feedback

Description

This variable determines 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.

The one exception to sequential propagation occurs when dealing with sequential integrated clock gating cells. These types of ICG cells will only propagate logic values when the *case_analysis_propagate_through_icg* variable is set to *true*.

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

```
printvar case_analysis_sequential_propagation

echo $case_analysis_sequential_propagation
```

See Also

- printvar
- set_case_analysis
- case_analysis_propagate_through_icg

---

case_analysis_with_logic_constants

Enables constant propagation when set to true, even if a design contains only logic constants.

Data Types

Boolean

Default true

Group

timing

Description

When set to true (the default), this variable enables constant propagation, even if a design contains only logic constants. When set to false, constant propagation is not performed unless a *set_case_analysis* command is specified. The *disable_case_analysis* variable overrides the *case_analysis_with_logic_constants* variable. If the *disable_case_analysis* variable is set, no constants are propagated.

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

Feedback

See Also

- remove_case_analysis
- report_case_analysis
- set_case_analysis
- disable_case_analysis

ccd_hold_control_effort

Set the effort level to control hold degradation during ccd setup optimization.

Data Types

String

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, and DC Explorer

high in Design Compiler NXT non-topographical mode and Design Compiler NXT topographical mode

Description

This application variable specifies the effort level for hold degradation allowed during CCD optimization. Higher effort level will result in better hold timing during CCD optimization but setup timing improvement during CCD can reduce.

Valid values are

- *low*
- *high*

ccd_ignore_ports_for_boundary_identification

Specifies the ports to exclude during boundary identification. This application variable is honored only when the *ccd_optimize_boundary_timing* application variable is *false*.

Data Types

string

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, and DC Explorer

"" in Design Compiler NXT non-topographical mode and Design Compiler NXT topographical mode

Feedback

Description

The *ccd_ignore_ports_for_boundary_identification* application variable specifies the ports to exclude when concurrent clock and data optimization (CCD) performs boundary identification. Typically you would specify high-fanout ports, such as scan enable and reset ports, to prevent large numbers of flip-flops from being identified as boundary flip-flops.

The tool performs boundary identification only when the *ccd_optimize_boundary_timing* application variable is *false*. If the *ccd_optimize_boundary_timing* application variable is *true*, the tool ignores the setting of the *ccd_ignore_ports_for_boundary_identification* application variable.

Examples

The following examples ignore the dddi[25] and clk1 ports during boundary identification.

```
prompt> set ccd_ignore_ports_for_boundary_identification "dddi[25] clk1"
ccd_ignore_ports_for_boundary_identification {dddi[25] clk1}

prompt> set ccd_ignore_ports_for_boundary_identification \\
                [list dddi[25] clk1]
ccd_ignore_ports_for_boundary_identification {{dddi[25]} clk1}
```

The following example ignores the input ports during boundary identification.

```
prompt> set ccd_ignore_ports_for_boundary_identification \\
                  [get_object_name [get_ports -filter "direction==in"]]
ccd.ignore_ports_for_boundary_identification {rst clk1 clk2 clk3 clk4
{sel[2]} {sel[1]} {sel[0]} {di[63]} {di[62]} {di[61]} {di[60]} ... }
```

See Also

- ccd_optimize_boundary_timing

---

ccd_ignore_scan_reset_for_boundary_identification

Specifies to exclude scan/reset connectivity during boundary identification. This application option is honored only when the *ccd_optimize_boundary_timing* application option is *false*.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, and DC Explorer

false in Design Compiler NXT non-topographical mode and Design Compiler NXT topographical mode

Feedback

Description

The *ccd_ignore_scan_reset_for_boundary_identification* application option is used to ignore the connectivities on scan/reset pins when concurrent clock and data optimization (CCD) performs boundary flop identification.

The tool performs boundary flop identification only when the *ccd_optimize_boundary_timing* application variable is false. If the *ccd_optimize_boundary_timing* application variable is true, the tool ignores the setting of the *ccd_ignore_scan_reset_for_boundary_identification* application variable.

See Also

- ccd_optimize_boundary_timing

---

ccd_max_postpone

Specifies the maximum postpone value adjusted during concurrent clock and data (CCD) optimization.

Data Types

float

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, and DC Explorer

0 in Design Compiler NXT non-topographical mode and Design Compiler NXT topographical mode

Description

CCD optimization adjusts clock latency to improve timing and power.

By default, there is no limit to reduce and increase clock latency during CCD optimization. To limit the value from adjusting during CCD optimization in Design Compiler NXT in topographical mode with physical guidance mode only, use the following application variables: - *ccd_max_prepone*: Limits reducing the value of clock latency. - *ccd_max_postpone*: Limits increasing the value of clock latency.

Specify a positive value in design units with the *ccd_max_prepone* and *ccd_max_postpone* application variables.

Examples

The following setting ensures that the maximum postpone value allowed during CCD optimization is 0.1 ns (design units in 1ns).

```
prompt> set_app_var ccd_max_postpone 0.1
```

Feedback

See Also

- ccd_max_prepone

---

ccd_max_prepone

Specifies the maximum prepone value adjusted during concurrent clock and data (CCD) optimization.

Data Types

float

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, and DC Explorer

0 in Design Compiler NXT non-topographical mode and Design Compiler NXT topographical mode

Description

CCD optimization adjusts clock latency to improve timing and power.

By default, there is no limit to reduce and increase clock latency during CCD optimization. To limit the value from adjusting during CCD optimization in Design Compiler NXT in topographical mode with physical guidance mode only, use the following application variables: - *ccd_max_prepone*: Limits reducing the value of clock latency. - *ccd_max_postpone*: Limits increasing the value of clock latency.

Specify a positive value in design units with the *ccd_max_prepone* and *ccd_max_postpone* application variables.

Examples

The following setting ensures that the maximum prepone value allowed during CCD optimization is 0.1 ns (design units in 1ns).

```
prompt> set_app_var ccd_max_prepone 0.1
```

See Also

- ccd_max_postpone

---

ccd_optimize_boundary_timing

Controls whether concurrent clock and data optimization (CCD) optimizes boundary timing paths.

Feedback

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, and DC Explorer

true in Design Compiler NXT non-topographical mode and Design Compiler NXT topographical mode

Description

The *ccd_optimize_boundary_timing* application variable controls whether CCD performs optimization on boundary timing paths.

By default (*true*), CCD optimizes boundary timing paths. In this case, all flip-flops in the block are available for CCD. The tool does not perform boundary identification and ignores the setting of the *ccd_ignore_ports_for_boundary_identification* and *ccd_ignore_scan_reset_for_boundary_identification* application variables.

To disable optimization on boundary timing paths, such as I/O paths, set the *ccd_optimize_boundary_timing* application variable to *false*. When this option is *false*,

- Only internal flip-flops are available for CCD

  The tool performs boundary identification to differentiate between the boundary and internal flip-flops. If your design contains ports that connect to a large number of flip-flops, such as scan enable and reset ports, you might want to exclude these ports from the boundary identification; otherwise, most of the flip-flops are considered boundary flip-flops and the CCD scope is very limited. To specify the ports to exclude, set the *ccd_ignore_ports_for_boundary_identification* application variable; If you specify *ccd_ignore_scan_reset_for_boundary_identification* application variable, all of the scan/reset ports will not to be considered.

- The tool does not adjust the latencies of the boundary flip-flops

  Note that setting the *ccd_optimize_boundary_timing* application variables to *false* prevents CCD from optimizing most of the clock tree.

See Also

- ccd_ignore_ports_for_boundary_identification
- ccd_ignore_scan_reset_for_boundary_identification

---

ccd_respect_cts_fixed_balance_pins

Used to control CCD engine not to optimize user specified sinks pins

Feedback

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, and DC Explorer

false in Design Compiler NXT non-topographical mode and Design Compiler NXT topographical mode

Description

This application variable is used to exclude the sink pins with attribute *cts_fixed_balance_pin* during CCD optimization and the tool will not adjust the latencies of these pins.

The valid values are

- *false* CCD in compile is free to adjust the latencies of the sink pins and does not honor the cts_fixed_balance_pin attribute on a pin
- *true* CCD in compile will not adjust the latencies of the sink pins with attribute cts_fixed_balance_pin. But the latencies to the sink pins with attribute cts_fixed_balance_pin can still change because the clock cells in the clock path to these sink pins can still be optimized by CCD when skewing other sink pins

Examples

```
The following examples control CCD engine to avoid to adjust latency for
 sinks pins with attribute cts_fixed_balance_pin:
prompt> set ccd_respect_cts_fixed_balance_pins true

The flop clock pins that need not be skewed by CCD should have the
 following setting:
set ff_cp [get_pins <flop clock pins>]
set_attribute $ff_cp cts_fixed_balance_pin true
```

See Also

- set_attribute
- get_attribute

---

ccd_skip_path_groups

Skip the path groups for concurrent clock and data (CCD) optimization.

Data Types

string

Feedback

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, and DC Explorer

"" in Design Compiler NXT non-topographical mode and Design Compiler NXT topographical mode

Description

This option accepts a list of path groups and CCD will skip these path groups during critical endpoint selection for optimization. This app option does not skip optimization on data path on the specified path groups.

---

ccd_write_script_sdc_add_offset_to_latency

Specifies how to write out clock latency offset constraints that are generated during concurrent clock and data optimization (CCD). This application variable is honored by the *write_script* and *write_sdc* commands.

Data Types

Boolean

Default false

Description

The *ccd_write_script_sdc_add_offset_to_latency* application variable specifies how to write out the clock latency offset constraints that are generated during concurrent clock and data optimization (CCD) in Design Compiler NXT in topographical mode. However, latency offset is not a SDC standard. Therefore, the *write_sdc* command ignores the latency offset constraints by default and the *write_script* command writes the latency offset constraints separately.

When you set the *ccd_write_script_sdc_add_offset_to_latency* variable to *true*,

1. The *write_sdc* and *write_script* commands add the latency offset value to latency value. This latency value is calculated from the user-defined latency constraints specified with the *set_clock_latency* command without *-offset* option.
2. The total latency value may be written out as latency value by *set_clock_latency* command without *-offset* option. It may overwrites the existing user-defined clock latency constraints.

Feedback

Examples

The following examples show the output of clock latency constraints in a file, written by the *write_sdc* and *write_script* command:

The default output of the *write_sdc* command:

```
set_clock_latency -max 1.01 -clock [get_clocks CLK1] [get_pins fd/CP]
```

The default output of the *write_script* command:

```
set_clock_latency -max 1.01 -clock [get_clocks CLK1] [get_pins fd/CP]
set_clock_latency -offset -max 0.03 -clock [get_clocks CLK1] [get_pins
 fd/CP]
```

When the Tcl value is true,

The output of the *write_sdc* command:

```
set_clock_latency -max 1.04 -clock [get_clocks CLK1] [get_pins fd/CP]
```

The output of the *write_script* command:

```
set_clock_latency -max 1.04 -clock [get_clocks CLK1] [get_pins fd/CP]
```

See Also

- write_script
- write_sdc

---

change_names_bit_blast_negative_index

Bit-blast the bus if any bit of it is negative.

Data Types

Boolean

Default false

Group

none

Description

When this variable is set to true, *change_names* bit-blasts the bus if any bit is negative. Otherwise, *change_names* shifts the negative range to the positive range starting at 0. The default value is false.

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

See Also

- change_names
- define_name_rules

change_names_dont_change_bus_members

Controls how the *change_names* command modifies the names of bus members.

Data Types

Boolean

Default false

Description

This variable is for the *change_names* command, and affects bus members only of bused ports or nets. When false (the default), *change_names* gives bus members the base name from their owning bus. For example, if BUS A has range 0 to 1 with the first element NET1 and the second element NET2, *change_names* changes NET1 to A[0] and NET2 to A[1]. When this variable is set to true, *change_names* does not change the names of bus members, so that NET1 and NET2 remain unchanged.

This variable also applies to *-special* rules, but has no effect if the name is changed by other rules that have higher priority than *-special* when *-special* rules are used; such as *-equal_ports_nets* and *-case_insensitive*. For more information, see the APPLYING NAME RULES section of the *define_name_rules* man page.

To determine the current value of this variable, use the *printvar change_names_dont_change_bus_members* command. For a list of all system variables and their current values, use the *print_variable_group system* command.

See Also

- change_names
- define_name_rules

check_design_allow_inconsistent_input_port

Specifies the severity level to apply when the tool finds a port that is not being used in accordance with its stated direction.

Feedback

Data Types

Boolean

Default false

Group

none

Description

When this variable is set to true, the *compile* and *check_design* commands issue warnings when ports that are not being used in accordance with their stated direction exist in the design. The command continues running after issuing a warning.

When the variable is set to false, the *compile* and *check_design* commands report errors when any such ports are detected. When the *compile* command encounters an error, it quits and returns a status of 0. When the *check_design* command encounters an error, it continues to run without any change in return status.

The default value of this variable is false.

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

See Also

- check_design
- compile
- LINT-69

---

check_design_allow_inconsistent_output_port

Specifies the severity level to apply when the Design Compiler tool finds an output port direction that is not used in accordance with its stated direction.

Data Types

Boolean

Default true

Group

none

Feedback

Description

When this variable is set to true, the *compile* and *check_design* commands issue warning messages when an output port direction is not used as specified in the design. The command continues running after issuing a warning message.

When the variable is set to false, the *compile* and *check_design* commands report error messages when any output ports are detected. When the *compile* command encounters an error, it quits and returns a status of 0. When the *check_design* command encounters an error, it continues to run without any change in return status.

The default is true.

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

See Also

- check_design
- compile
- LINT-70
- LINT-68

---

check_design_allow_multiply_driven_nets_by_inputs_and_outputs

Controls whether the tool checks for input ports that connect to multiply driven nets whose drivers include an output port or pin.

Data Types

Boolean

Default false

Group

none

Description

When this variable is set to *true*, the *compile* and *check_design* commands skip the checking of input ports that connect to multiply driven nets whose drivers include an output port or pin.

When the variable is set to *false*, the tool reports errors when input ports are connected to multiply driven nets whose drivers include an output port or pin. When the *compile*

command encounters an error, it quits and returns a status of 0. When the *check_design* command encounters an error, it continues to run without any change in return status.

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

See Also

- check_design
- compile

---

check_design_allow_non_tri_drivers_on_tri_bus

Specifies the severity level to apply when the tool finds three-state buses with non-three-state drivers in the design.

Data Types

Boolean

Default true

Group

none

Description

When this variable is set to true, the *compile* and *check_design* commands issue warnings when three-state buses with non-three-state drivers exist in the design. The command continues running after a warning is issued.

When the variable is set to false, the *compile* and *check_design* commands report errors on three-state buses. When the *compile* command encounters an error, it quits and returns a status of 0. When the *check_design* command encounters an error, it continues to run without any change in return status.

The default value of this variable is true.

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

See Also

- check_design
- compile

Feedback

check_design_allow_unknown_wired_logic_type

Specifies the severity level to apply when the tool finds nets with multiple drivers of the unknown wired-logic type.

Data Types

Boolean

Default true

Group

none

Description

When this variable is set to *true*, the *compile* and *check_design* commands report a warning for nets with multiple drivers of the unknown wired-logic type. When the variable is set to *false*, the commands report an error.

When the *compile* command encounters an error, it quits and returns a status of 0. When the *check_design* command encounters an error, it continues to run without any change in return status.

The default value of this variable is *true*.

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

See Also

- check_design
- compile

check_design_check_for_wire_loop

Controls whether the tool checks for wire loops that have a timing loop with no cells in it.

Data Types

Boolean

Default true

Group

none

Feedback

Description

When this variable is set to true, the *compile* and *check_design* commands issue When the *compile* command encounters an error, it quits and returns a status of 0. When the *check_design* command encounters an error, it continues to run without any change in return status.

When the variable is set to false, the tool skips the checking of wire loops in the design.

The default value of this variable is true.

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

See Also

- check_design
- compile

---

check_design_check_multidriven_output_ports

Controls whether the tool checks for output ports that connect to multi-driver nets.

Data Types

Boolean

Default false

Group

none

Description

When this variable is set to *true*, the *check_design* command checks for output ports that connect to multi-driver nets.

When the variable is set to *false*, the tool does not check for output ports that connect to multi-driver nets.

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

See Also

- check_design

Feedback

check_error_list

Specifies the error codes for which the *check_error* command checks.

Data Types

list

Default CMD-004 CMD-006 CMD-007 CMD-008 CMD-009 CMD-010 CMD-011 CMD-012 CMD-014 CMD-015 CMD-016 CMD-019 CMD-026 CMD-031 CMD-037 DB-1 DCSH-11 DES-001 ACS-193 FILE-1 FILE-2 FILE-3 FILE-4 LINK-7 LINT-7 LINT-20 LNK-023 OPT-100 OPT-101 OPT-102 OPT-114 OPT-124 OPT-127 OPT-128 OPT-155 OPT-157 OPT-181 OPT-462 UI-11 UI-14 UI-15 UI-16 UI-17 UI-19 UI-20 UI-21 UI-22 UI-23 UI-40 UI-41 UID-4 UID-6 UID-7 UID-8 UID-9 UID-13 UID-14 UID-15 UID-19 UID-20 UID-25 UID-27 UID-28 UID-29 UID-30 UID-32 UID-58 UID-87 UID-103 UID-109 UID-270 UID-272 UID-403 UID-440 UID-444 UIO-2 UIO-3 UIO-4 UIO-25 UIO-65 UIO-66 UIO-75 UIO-94 UIO-95 EQN-6 EQN-11 EQN-15 EQN-16 EQN-18 EQN-20

Description

This variable specifies the error codes for which the *check_error* command checks. The *check_error* command returns a 1 if any of the specified error codes have been generated by a previous command in the current session.

You can use this capability to stop batch jobs in which the specified error codes occur.

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

See Also

- check_error

collection_result_display_limit

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

Data Types

integer

Default 100

Description

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

Feedback

When a command (for example, *add_to_collection*) is issued at the command prompt, its result is implicitly queried, as though the *query_objects* command 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 the *printvar collection_result_display_limit* command.

See Also

- collections
- printvar
- query_objects

---

command_log_file

The *command_log_file* variable is obsolete. Use the *sh_command_log_file* variable instead.

---

company

Specifies the name of the company where Synopsys software is installed. The company name is displayed on the schematics.

Data Types

string

Default ""

Description

This variable specifies the name of the company where Synopsys software is installed. The company name is displayed on the schematics.

To determine the current value of this variable use the *printvar company* command. For a list of all system variables and their current values, use the *print_variable_group system* command.

---

compatibility_version

Sets the default behavior of the system to be the same as the Synopsys software version specified in the variable.

Feedback

Data Types

string

Default current release

Description

This variable sets the default behavior of the system to be the same as the Synopsys software version specified in the variable. This setting provides compatibility for script command files written in previous software versions. The scripts are run on the current version of the software, so results are usually better. However, the script performs the same default actions here as it did on the specified software version.

To determine the current value of this variable use the *printvar compatibility_version* command. For a list of all system variables and their current values, use the *print_variable_group system* command.

---

compile_adjust_max_processes_used

Enables automatic adjustment of number of multiprocesses in *compile*, *compile_ultra*, *parallel_execute*, and *redirect -bg* commands.

Data Types

Integer

Default 1

Description

By default, the *compile* or *compile_ultra* command runs new processes in parallel. The number of cores to run processes is controlled by the *set_host_options -max_cores* command.

When you set the variable to *1* (the default) and if there are insufficient resources to run new processes in parallel, the *compile* or *compile_ultra* command adjusts the number of cores used to proceed, instead of terminating. However, the command continues to execute if and only if one new process succeeds.

Similarly, the *parallel_execute* command reduces the number of processes to be used and continues to execute in the main process if the first new process fails. The *Redirect -bg* command executes in the main process if new processes fail.

Feedback

The tool behaves as follows based on the *compile_adjust_max_processes_used* variable settings:

- *1* (the default): The tool automatically adjusts the number of cores during multiprocessing if the *overcommit_memory* parameter is set to *2* (no overcommit).
- *0*: The tool does not adjust the number of cores.
- *2*: the tool adjusts the number of cores irrespective of the *overcommit_memory* parameter settings.

See Also

- UIO-306
- UIO-307

---

compile_advanced_fix_multiple_port_nets

Controls whether the *compile* or *compile_ultra* command fixes multiple port nets using rewiring and buffering.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

true in DC Explorer

Description

When the value is *true*, the *compile* or *compile_ultra* command fixes multiple port nets using rewiring and buffering. Where possible, all multiple port nets and constant-driven ports are fixed by changing the connections elsewhere in the hierarchy. Any remaining multiple port nets and constant ports are fixed using buffering. Note: The variable is set to on-by-default only in the physical guidance flow, when you use the *compile_ultra* command with the -spg option. If you do not want to use the feature in the physical guidance flow, explicitly set the compile_advanced_fix_multiple_port_nets variable to *false*

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

```
prompt> printvar compile_advanced_fix_multiple_port_nets
```

See Also

- set_fix_multiple_port_nets

Feedback

compile_allow_dw_hierarchical_inverter_opt

Allows the phase inversion boundary optimization to be applied to DesignWare components.

Data Types

Boolean

Default false

Description

The *compile_allow_dw_hierarchical_inverter_opt* variable allows the phase inversion boundary optimization to be applied to DesignWare components, when boundary optimization in general is permitted on the component.

The default value of this variable is false, which prevents the *compile* command from changing the phase of DesignWare component output signals. Other boundary optimizations are still attempted.

Setting the variable to true enables the phase inversion on DesignWare component outputs. Note that the true setting may interfere with formal verification of the design.

See Also

- compile

compile_assume_fully_decoded_three_state_busses

Specifies whether the *compile* and *translate* commands can assume that three-state buses are fully decoded.

Data Types

Boolean

Default false

Description

When this variable value is set to true, the *compile* and *translate* commands assume that three-state buses are fully decoded, and therefore can be replaced by multiplexed buses when mapping to a library that contains no three-state cells. The default value is false.

To determine the current value of this variable, use the *printvar compile_assume_fully_decoded_three_state_busses* command. For a list of compile variables and their current values, use the *print_variable_group compile* command.

Feedback

compile_auto_ungroup_area_num_cells

Defines the minimum number of child cells a design hierarchy must have so that the *compile -auto_ungroup area* command does not ungroup the hierarchy.

Data Types

integer

Default 30 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

The *compile_auto_ungroup_area_num_cells* variable defines the minimum number of child cells a design hierarchy must have so that the *compile -auto_ungroup area* command does not ungroup the hierarchy. The default value for this variable is 30. By default, the threshold check is done only on the child cells of its immediate hierarchy. You can enable the threshold check to include all of the leaf cells of this design hierarchy (for example, both the child cells of its immediate hierarchy and all of its subdesigns) by setting the *compile_auto_ungroup_count_leaf_cells* variable to true.

To determine the current value of this variable, use the *printvar compile_auto_ungroup_area_num_cells* command. For a list of all compile variables and their current values, use the *print_variable_group compile* command.

See Also

- compile
- compile_auto_ungroup_count_leaf_cells
- compile_auto_ungroup_override_wlm

compile_auto_ungroup_count_leaf_cells

Determines if the number of leaf cells should be counted or the number of child cells in the immediate hierarchy should be counted to compare against the value of the *compile_auto_ungroup_area_num_cells* variable in area-based auto-ungrouping.

Data Types

Boolean

Feedback

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

The *compile_auto_ungroup_count_leaf_cells* variable determines the way the tool counts the number of child cells of a design hierarchy. When this variable is set to false, the default value, only child cells in the immediate hierarchy are counted; the number of child cells in its subdesigns is considered.

When this variable is set to true, the leaf cells of a design hierarchy (for example, both the child cells of its immediate hierarchy and all its subdesigns) are considered. The number of child cells of a design hierarchy is compared against the value of *compile_auto_ungroup_area_num_cells* in area-based auto-ungrouping to determine if a design hierarchy can be considered as a candidate for ungrouping.

Because of this, for the same design hierarchy and same variable settings for auto-ungroup, a certain design hierarchy may be ungrouped if *compile_auto_ungroup_count_leaf_cells* is set to false, but may not be ungrouped if it is set to true. For example, assume design hierarchy A has 20 immediate child cells and a subdesign B, and subdesign B has another 40 immediate child cells with no more subdesigns. In a compile flow where *compile -auto_ungroup area* is used and the *compile_auto_ungroup_area_num_cells* variable is set to 30, if *compile_auto_ungroup_count_leaf_cells* is set to false, design hierarchy A will be considered as a candidate for auto-ungrouping because it has 20 < 40 child cells in its immediate hierarchy. However, if *compile_auto_ungroup_count_leaf_cells* is set to true, since design A has 20+40 = 60 > 40 leaf cells, it will be not be considered as a candidate for auto-ungrouping.

To determine the current value of this variable, use the *printvar compile_auto_ungroup_count_leaf_cells* command. For a list of all compile variables and their current values, use the *print_variable_group compile* command.

See Also

- compile
- compile_auto_ungroup_area_num_cells

compile_auto_ungroup_override_wlm

Specifies whether the compiler considers a cell instance for automatic ungrouping, if the cell's wire load model differs from that of its parent.

Feedback

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable specifies whether the compiler considers a cell instance for automatic ungrouping, if the cell's wire load model is different from the wire load model of its parent cell. The default value is false, which means that the cell instance is not considered for automatic ungrouping should its wire load model differ from that of its parent.

If you set the value of this variable as true, the ungrouped child cells of the named cell instance inherit the wire load model of its parent. A result of setting this variable to true is that the tool might use more pessimistic wire load models for the child cells of this cell instance. This in turn might offset the delay improvement from the *compile -auto_ungroup area* or *-compile -auto_ungroup delay* command and lead to seemingly worse timing behavior for the design.

To determine the current value of this variable, use the *printvar compile_auto_ungroup_override_wlm* command. For a list of all compile variables and their current values, use the *print_variable_group compile* command.

See Also

- compile
- compile_auto_ungroup_area_num_cells

---

compile_automatic_clock_phase_inference

Specifies the method used to determine the clock phase during sequential mapping.

Data Types

string

Default strict

Description

When this variable is set to strict, the *compile* command attempts to determine the desired clock phase for each unmapped register, and does not allow opposite phase devices to be used in constructing registers.

When the value is set to relaxed, *compile* allows the implementation of an opposite phase device for a register only if there is no other way to implement that register.

When set to none, *compile* ignores the clock phase during sequential mapping. The default value is *strict*.

To determine the current value of this variable, use the *printvar compile_automatic_clock_phase_inference* command. For a list of compile variables and their current values, use the *print_variable_group compile* command.

---

compile_cell_density_aware_optimization

Enable cell density awareness, during delay optimizations in *DCNXT compile_ultra -spg*.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, and DC Explorer

false in Design Compiler NXT non-topographical mode and Design Compiler NXT topographical mode

Description

During Post placement delay optimization, *Design Compiler* can clump/over-lap critical cells in the design while improving WNS, TNS. This could impact the correlation to post legalization result in IC Compiler II. When you set the *compile_cell_density_aware_optimization* variable to *true*, Design Compiler identifies and reduces cell density violations, while performing Delay Optimizations. This lessens the impact on correlation to post legalization of IC Compiler II. The design will have reduced density violations when this variable is enabled, but not always a zero violation. This is not a placement feature and none of the placer density variables will affect it.

Examples

The following example shows how to enable and disable this feature for a design:

```
prompt> set compile_cell_density_aware_optimization true
prompt> set compile_cell_density_aware_optimization false
```

See Also

- compile_ultra
- compile_cell_density_aware_optimization_threshold

Feedback

compile_cell_density_aware_optimization_threshold

The value you specify sets the threshold for how tightly the cells are allowed to clump during delay optimizations.

Data Types

float

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, and DC Explorer

1.15 in Design Compiler NXT non-topographical mode and Design Compiler NXT topographical mode

Description

When *compile_cell_density_aware_optimization* variable is set to *true*, the value of the *compile_cell_density_aware_optimization_threshold* variable determines which regions of physical design are considered to be *dense*. During delay optimizations, Design Compiler tries to control the dense regions by preventing cell clumping above this threshold.

Examples

The following example sets the threshold to 1.5.

prompt> *set compile_cell_density_aware_optimization_threshold 1.5*

See Also

- compile_cell_density_aware_optimization

compile_checkpoint_phases

Determines whether checkpoints are generated during execution of the *compile* command.

Data Types

Boolean

Default false in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

When true, checkpoints automatically between each phase of *compile*. The default is false.

Feedback

To determine the current value of this variable, type *printvar compile_checkpoint_phases*. For a list of all *compile* variables and their current values, type *print_variable_group compile*.

---

compile_clock_cells_name_prefix

Cells on clock network will be prefixed with user given string.

Data Types

string

Default true

Group

none

Description

This variable 'compile_clock_cells_name_prefix' allows user to prefix cells on clock network with provided value.

If user do not want cells on clock network to be prefixed, set this variable is to *""*.

This feature only applies when user uses 'set_target_library_subset -clock_path'

The default value of compile_clock_cells_name_prefix is *CLOCK_U*.

See Also

- set_target_library_subset

---

compile_clock_gating_through_hierarchy

Controls whether the *compile* or *compile_ultra* command with the *-gate_clock* option performs clock gating through hierarchy boundaries.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

When this variable is set to *true*, *compile -gate_clock* and *compile_ultra -gate_clock* are allowed to use one clock gate to gate registers in different hierarchical cells. This can increase the number of clock gating opportunities and reduce the number of clock gates.

When the value is *false* (the default), the clock gating is only performed in such a way that clock gates are in the same hierarchy cell as all registers gated by them.

A clock gating cell will not be modified or removed if it or its parent hierarchical cell is marked *dont_touch* with the *set_dont_touch* command; it will not be modified by global clock gating optimization.

To determine the current value of this variable, use the *printvar compile_clock_gating_through_hierarchy* command. For a list of all compile variables and their current values, use *print_variable_group compile*.

This variable is not supported in DC Explorer.

---

compile_create_wire_load_table

Controls the type of wire load model generated by the *create_wire_load* command.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable is used to control the type of wire load model generated by the *create_wire_load* command. The default setting of this variable is false and the wire load models generated are in the wire_load format. It includes resistance, capacitance, area slope coefficients, and fanout_length (fanout, length, average_cap, std_dev, and points)

If this variable is set to true the wire load models generated are in the wire_load_table format. The resistance is calculated explicitly for each fanout using the back-annotated values and the tree type from the current operating condition set on the design.

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

For a list of all *compile* variables and their current values, use the *print_variable_group compile* command. For a list of all *links_to_layout* variables and their current values, type *print_variable_group links_to_layout*.

See Also

- compile

---

compile_delete_unloaded_sequential_cells

Controls whether the *compile* command deletes unloaded sequential cells.

Data Types

Boolean

Default true

Description

By default, the *compile* command deletes unloaded sequential cells. To retain such cells, set the *compile_delete_unloaded_sequential_cells* variable to false.

During compile, if a design contains sequential cells that do not drive loads, the logic driven by that sequential cell might be optimized away, resulting in an inferred no-load cell,or no path to a primary output.

See Also

- compile

---

compile_dft_log_print_mis_info

If this is true, info about multi-input switching cells decomposition will be displayed in the log file. Please note that, this will be supported only if MIS cells decomposition is enabled using *compile_enable_mis_cells_decomposition* or *dft_enable_mis_cells_decomposition* variable.

Data Types

Boolean

Default false

Description

If this is true, info about multi-input switching cells decomposition will be displayed in the log file using INFO-123 messages. This helps to display the reason for skipping

certain cells from decomposition. Please note that, this will be supported only if MIS cells decomposition is enabled using *compile_enable_mis_cells_decomposition* or *dft_enable_mis_cells_decomposition* variable.

Examples

The following examples show how to enable and disable printing of information about skipped cells during decomposition of MIS cells:

```
prompt> set compile_dft_log_print_mis_info true
prompt> set compile_dft_log_print_mis_info false
```

See Also

- compile_enable_mis_cells_decomposition
- dft_enable_mis_cells_decomposition
- report_mis_violation_summary

---

compile_disable_hierarchical_inverter_opt

Controls whether inverters can be moved across hierarchical boundaries during boundary optimization.

Data Types

Boolean

Default false

Description

This variable affects the behavior of the boundary optimization feature in the *compile* command. By default, boundary optimization attempts to push inverters across hierarchies to improve the optimization cost of the design. However, if the *compile_disable_hierarchical_inverter_opt* variable is set to true, boundary optimization will not move inverters across hierarchical boundaries even if doing so could improve the design.

To determine the current value of this variable, use the *printvar compile_disable_hierarchical_inverter_opt* command. For a list of all *compile* variables and their current values, use the *print_variable_group compile* command.

Feedback

See Also

- compile
- set_boundary_optimization
- port_complement_naming_style

---

compile_dont_touch_annotated_cell_during_inplace_opt

Controls whether cells that have annotated delays can be optimized.

Data Types

Boolean

Default false in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

When true, *reoptimize_design -in_place* and *compile -in_place* disallow swapping cells that have annotated delays. When false (the default value), *reoptimize_design -in_place* and *compile -in_place* allow annotated cells to be swapped for cells without annotated delay.

This variable is used to run *reoptimize_design -in_place* and *compile -in_place* with annotated cell and net delays and allowing only buffers to be inserted. When this variable is true, Design Compiler considers all cells with annotated delays as *dont_touch*. This allows avoiding exchanging a cell with annotated delays for a cell with lower estimated delays but higher real delays.

To determine the current value of this variable, use *printvar compile_dont_touch_annotated_cell_during_inplace_opt*. For a list of all *compile* variables and their current values, use the *print_variable_group compile* command. For a list of all *links_to_layout* variables and their current values, use the *print_variable_group links_to_layout* command.

See Also

- compile

Feedback

compile_dont_use_dedicated_scanout

Controls whether optimizations use a scan cell's dedicated scan-out pin for functional connections.

Data Types

integer

Default 1

Description

Optimizations by *place_opt*, *clock_opt*, *route_opt* and *psyn_opt* do not use a scan cell's dedicated scan-out pin for functional connections. When this variable is set to 0, optimizations can use dedicated scan-out pins for functional connections.

Dedicated scan-out pins must be identified in the technology library using the *test_output_only* attribute. Contact your ASIC Vendor to ensure that dedicated scan-out pins are correctly modeled in the library that you are using.

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

compile_enable_async_mux_mapping

Controls whether the *compile* or *compile_ultra* command tries to preserve multiplexers in the fanin cone of asynchronous register pins.

Data Types

Boolean

Default true

Description

When this variable is set to *true*, the *compile* or *compile_ultra* command tries to preserve multiplexers in the fanin cone of asynchronous register pins. More specifically, if multiplexing logic (any MUX_OP or 2-input SELECT_OP) is found that is in the fanin of only asynchronous register pins (clock, and asynchronous set and reset), then this logic is mapped to a multiplexer, and a size-only attribute is set on the multiplexer. If the MUX_OP or SELECT_OP cell is specified as dont_touch, Design Compiler does not map the cell to a multiplexer. The purpose of this is to reduce the occurrence of glitches on asynchronous register pins. The default value of this variable is true.

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

For a list of all compile variables and their current values, use *print_variable_group compile*.

See Also

- hdlin_mux_size_only

---

compile_enable_ccd

Enables concurrent clock and data optimization (CCD) during compile.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, and DC Explorer

false in Design Compiler NXT non-topographical mode and Design Compiler NXT topographical mode

Description

With this application variable set to true, concurrent clock and data optimization (CCD) is enabled during full compile as well as incremental compile which does the useful skew computation. The useful skew computation incrementally updates the existing user-specified offsets (both balance points and clock latencies).

The magnitude of skew offsets added by compile CCD is bounded by the *ccd_max_postpone* and the *ccd_max_prepone* application variables.

The general recommendation is to align with the CCD setting during place_opt.

See Also

- compile
- ccd_max_postpone
- ccd_max_prepone

---

compile_enable_constant_propagation_with_no_boundary_opt

Controls whether the constants are propagated across hierarchies when boundary optimization is disabled.

Data Types

Boolean

Feedback

Default true

Description

When this variable is set to *true* (the default), the tool propagates constants across design hierarchies and simplifies the logic even when boundary optimization is disabled by setting the *boundary_optimization* attribute to *false* or using the *-no_boundary_optimization* option with the *compile_ultra* command.

When the variable is set to *false*, the tool does not propagate constants across hierarchical boundaries.

The variable has no effect on other optimizations, such as the propagation of equal and opposite information, the propagation of unconnected port information, and phase inversion.

To stop constant propagation through specific design hierarchies, use the *set_compile_directives* command with the *-constant_propagation* option set to *false* on a specific hierarchical pin or a list of hierarchical pins. When you do this, the tool sets a *const_prop_off* attribute on the specified pins and disables constant propagation through them.

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

printvar compile_enable_constant_propagation_with_no_boundary_opt

See Also

- compile_ultra
- set_boundary_optimization
- set_compile_directives

---

compile_enable_enhanced_leakage_optimization

Enables enhanced leakage optimization in *compile_ultra*.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Feedback

Description

When you set the *compile_enable_enhanced_leakage_optimization* variable to *true*, the *compile_ultra* command runs a strategy intended to improve the leakage of the design, possibly at the cost of additional runtime and a change of qor trajectroy. Use the *compile_ultra* command without the variable as the default strategy, and enable this variable to fine tune the flow, as needed, on designs that require further leakage optimization. During optimization, DCNXT high effort flow configurations, enabled with *set_compile_power_high_effort*, will take precedence where available over enhanced leakage optimization configurations.

Examples

The following example shows how to enable and disable enhanced leakage optimization for a design:

```
prompt> compile_enable_enhanced_leakage_optimization true
prompt> compile_enable_enhanced_leakage_optimization false
```

See Also

- compile_ultra
- set_compile_power_high_effort

---

compile_enable_min_delay_fixing

Enables hold fixing in the DC Ultra wire load model flow when set to *true*.

Data Types

Boolean

Default false in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

When you set *compile_enable_min_delay_fixing* to *true* in DC Ultra wire load mode, Design Compiler performs minimum path, minimum capacitance, and minimum transition time fixing. However, be aware that enabling this variable can increase area and runtime.

The *compile_enable_min_delay_fixing* variable has no effect in DC Expert, Design Compiler in topographical mode, or Design Compiler Graphical flows. Hold fixing is not performed in these flows.

Feedback

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

```
prompt> printvar compile_enable_min_delay_fixing
```

compile_enable_mis_cells_decomposition

If this is true, combinational multi-input switching cells tied to same net will be decomposed as part of compile and incremental compile(s). This feature is supported in DCG and DCNXT only.

Data Types

Boolean

Default false

Description

The variable *compile_enable_mis_cells_decomposition* is used to enable or disable decomposition of combinational MIS cells tied to same net in the subsequent compile command(s). This feature is supported in DCG and DCNXT only.

Examples

The following examples show how to enable and disable decomposition of MIS cells during compile:

```
prompt> set compile_enable_mis_cells_decomposition true
prompt> set compile_enable_mis_cells_decomposition false
```

See Also

- report_mis_violation_summary
- compile_dft_log_print_mis_info

compile_enable_multibit_rewiring_in_incremental

Data Types

Boolean

Default false

Description

When set to true before *compile_ultra -incremental*, and when the target library includes library cells with mixed drive properties, a new step will be enabled that will apply sizing with rewiring capabilities to the multi-bit registers of the netlist in order to save power when not all the slices of a multi-bit register are critical.

Feedback

See Also

- enable_enhanced_physical_multibit_banking
- compile_enable_physical_multibit_banking

---

compile_enable_physical_multibit_banking

Data Types

Boolean

Default false

Description

When set to true before main *compile_ultra*, the enhanced physical multibit banking engine will be executed after the second placement of the netlist during main *compile_ultra* in order to improve the QoR before incremental compile as compared to a standalone physical multibit banking.

See Also

- enable_enhanced_physical_multibit_banking

---

compile_enable_print_messages_for_redundant_registers

Enables printing of messages for redundant registers in the dc-shell.

Data Types

Boolean

Default false

Description

This variable enables the printing of messages for redundant registers in dc-compile. When the variable is set to true, the tool prints the name of register pins which, have become constant or unloaded due to redundancy removal optimiation, and could be optimized as constant/unloaded registers in later steps.

Examples

The following examples show how to enable and disable printing messages for redundant registers:

```
prompt> set compile_enable_print_messages_for_redundant_registers true
prompt> set compile_enable_print_messages_for_redundant_registers false
```

Feedback

See Also

- CGRR-001
- CGRR-002

---

compile_enable_register_merging

Controls whether the *compile* command should identify and merge equal and opposite registers.

Data Types

Boolean

Default true

Description

When this variable is set to *true* (the default), the *compile* command tries to identify and merge registers in the current design that are equal or opposite. This improves the area of the design.

When two registers are merged, the tool issues an OPT-1215 message.

Register merging can also be controlled by a design-by-design or a cell-by-cell basis by using the *set_register_merging* command. However, the *compile_enable_register_merging* variable takes precedence over the *set_register_merging* command.

Note: No register merging is performed if the *compile_enable_register_merging* variable is set to *false*.

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

For a list of all compile variables and their current values, use the *print_variable_group compile* command.

See Also

- set_register_merging
- compile_enable_register_merging_with_exceptions

---

compile_enable_register_merging_with_exceptions

Controls whether the *compile* command tries to identify and merge equal and opposite registers with path group exceptions.

Feedback

Data Types

Boolean

Default true in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

When this variable is set to *true*, the *compile* command tries to identify and merge registers in the current design that are equal or opposite even if they are with path group exceptions, as long as they are in the same path group. This improves the area of the design.

When this variable is set to *false* (the default), the *compile* command does not merge registers with path group exceptions.

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

For a list of all compile variables and their current values, use *print_variable_group compile*.

See Also

- set_register_merging
- compile_enable_register_merging

---

compile_enable_register_replication_without_reset_pin

Controls register replication of registers without reset pin with the *compile_ultra* and *compile* commands.

Data Types

Boolean

Default true

Description

When this variable is set to *false*, *compile* will not allow replication of registers without reset pin.

When this variable is set to *true* (default value), *compile* will allow replication of registers without reset pin, if it is a candidate for replication.

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

For a list of all compile variables and their current values, use *print_variable_group compile*.

See Also

- set_register_replication
- compile_register_replication

---

compile_enable_report_transformed_registers

Controls whether the register transformation are recorded and can be reported by the report_transformed_registers command

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, and DC Explorer

true in Design Compiler NXT non-topographical mode and Design Compiler NXT topographical mode

Description

When this variable is set to *false*, the recording of register transformations will be disabled and report_transformed_registers will not report the transformations. The default value of this variable is *true.*

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

```
prompt> printvar compile_enable_report_transformed
```

See Also

- report_transformed_registers

---

compile_enable_total_power_optimization

Controls DCNXT total power optimization feature.

This variable is supported only in DCNXT shell topographical mode for -spg flow.

Feedback

Data Types

Boolean

Default false

Description

This variable controls NXT total power optimization feature which includes both sizing and placement related total power optimizations. For the DCNXT total power optimization feature to work in MCMM designs, atleast one scenario must set *-setup true* and *-dynamic_power true*.

This variable is supported only in DCNXT shell topographical mode for -spg flow.

Allowed values are *false* and *true*.

When set to *false*, DCNXT optimizes only for leakage power.

When set to *true*, DCNXT optimizes for both leakage and dynamic power. Enabling DCNXT total power optimization features can lead to different QoR tradeoff behaviors than the ones seen with the default compile flow and hence, it can affect WNS and/or TNS and/or leakage power.

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

See Also

- compile_ultra
- set_scenario_options
- set_compile_power_high_effort

---

compile_enhanced_tns_optimization

Enables enhanced TNS optimization (ETO) for delay optimization engines with the *compile_ultra -spg* command in the Design Compiler NXT tool.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, and DC Explorer

false in Design Compiler NXT non-topographical mode and Design Compiler NXT topographical mode

Feedback

Description

When you set the *compile_enhanced_tns_optimization* variable to *true*, the *compile_ultra* command while running in *dcnxt_shell* runs a strategy intended to improve the TNS of the design, possibly at the cost of additional runtime, area, and worst negative slack (WNS).

Use the *compile_ultra* command without the variable as the default strategy, and enable the *compile_enhanced_tns_optimization* variable to fine-tune the flow as needed on designs that require further TNS optimization. This strategy performs a different method of selecting optimization candidates to achieve better TNS optimization.

To enable enhanced TNS optimization with the *compile_ultra* and *compile_ultra -incremental* commands, use the *set_app_var compile_enhanced_tns_optimization true* variable.

From R2020.09-SP1 *compile_enhanced_tns_optimization_in_incremental* is on by default when *compile_timing_high_effort_tns* is enabled and the user is running *compile_ultra -incremental*.

Examples

To enable enhanced TNS optimization only with the *compile_ultra* command and not with the *compile_ultra -incremental* command, set the variables as follows:

```
prompt> set_app_var  compile_enhanced_tns_optimization true
prompt> set_app_var  compile_enhanced_tns_optimization_in_incremental
 false
```

To disable enhanced TNS optimization during compile, set the variables as follows:

```
prompt> set_app_var  compile_enhanced_tns_optimization false
prompt> set_app_var  compile_enhanced_tns_optimization_in_incremental
 default
```

See Also

- compile_ultra
- compile_enhanced_tns_optimization_in_incremental
- compile_enhanced_tns_optimization_effort_level

---

compile_enhanced_tns_optimization_effort_level

Effort level parameter for *compile_enhanced_tns_optimization* for enhanced cell candidate selection for delay optimization engines in *DCNXT compile_ultra -spg*.

Data Types

String

Feedback

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, and DC Explorer

low in Design Compiler NXT non-topographical mode and Design Compiler NXT topographical mode

Description

When you meet the requirements for running and have enabled *compile_enhanced_tns_optimization* variable to *true*, large designs may benefit from setting *compile_enhanced_tns_optimization_effort_level* to *"medium"* or *"high"*, rather than the default of *"low"*. Adjust this variable to fine tune the flow, as needed, on large designs that require further tns optimization. Timing on smaller designs may be negatively impacted when running a higher effort level. Available effort options from lowest effort to highest effort are *"ultra_low"*, *"low"*, *"medium"*, *"high"*.

Examples

The following example shows how to set the parameter for enhanced tns optimization effort level for a design:

```
prompt> set compile_enhanced_tns_optimization_effort_level "ultra_low"
prompt> set compile_enhanced_tns_optimization_effort_level "low"
prompt> set compile_enhanced_tns_optimization_effort_level "medium"
prompt> set compile_enhanced_tns_optimization_effort_level "high"
```

See Also

- compile_ultra
- compile_enhanced_tns_optimization

---

compile_enhanced_tns_optimization_in_incremental

Enables enhanced TNS optimization (ETO) for delay optimization engines with the *compile_ultra -incremental -spg* command in the Design Compiler NXT tool.

This strategy performs a different method of selecting optimization candidates to achieve better TNS optimization while running in dcnxt_shell, possibly at the cost of additional runtime area and WNS.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, and DC Explorer

default in Design Compiler NXT non-topographical mode and Design Compiler NXT topographical mode

Description

With this variable set to *default*, the delay optimization engines strictly observe the value defined by the *compile_enhanced_tns_optimization* variable. The default setting also allows backward compatibility, if needed.

From R2020.09-SP1 *compile_enhanced_tns_optimization_in_incremental* is on by default when *compile_timing_high_effort_tns* is enabled and the user is running *compile_ultra -incremental*.

However, setting the *compile_enhanced_tns_optimization_in_incremental* variable to *true* or *false* allows more explicit control over enhanced TNS optimization in the *compile_ultra -incremental* flow step.

It is generally recommended

- To use the *compile_ultra -incremental* command without enhanced TNS optimization as the default strategy

  and

- To enable enhanced TNS optimization in the *compile_ultra -incremental* step to fine-tune the flow as needed on designs that require further TNS optimization

Depending on scripting and debugging requirements, enhanced TNS optimization can be enabled during the *compile_ultra -incremental* step (if required) in one of the following ways:

- Set the *compile_enhanced_tns_optimization* variable to *true* only before the *compile_ultra -incremental* step

  or

- Set the *compile_enhanced_tns_optimization_in_incremental* variable to *true* at any point in the flow either before the *compile_ultra* or *compile_ultra -incremental* step

Setting of the variables is not order dependent, as the tool uses the *compile_enhanced_tns_optimization* value for both compile and incremental compile, unless the *compile_enhanced_tns_optimization_in_incremental* variable is set to *true* or *false*.

Feedback

Examples

To enable enhanced TNS optimization only with the *compile_ultra -incremental* command and not with the *compile_ultra* command, set the variables as follows:

```
prompt> set_app_var  compile_enhanced_tns_optimization false
prompt> set_app_var  compile_enhanced_tns_optimization_in_incremental
 true
```

To disable enhanced TNS optimization during the compile and incremental compile steps, set the variables as follows:

```
prompt> set_app_var  compile_enhanced_tns_optimization false
prompt> set_app_var  compile_enhanced_tns_optimization_in_incremental
 default
```

See Also

- compile_ultra
- compile_enhanced_tns_optimization
- compile_enhanced_tns_optimization_effort_level

---

compile_final_drc_fix

Controls final DRC fixing stage of the compile command to fix DRC violations.

This variable is supported only in topographical mode.

Data Types

string

Default none

Description

This variable controls whether final DRC fixing stage is invoked or not to fix different type of DRC violations.

This variable is supported only in topographical mode.

Allowed values are *none* (the default), *transition*, *fanout*, *capacitance* and *all*.

When set to *none*, no DRC violation are fixed in final DRC fixing stage of compile command.

When set to *transition*, only max_transition violations are fixed in final DRC fixing stage of compile command.

Feedback

When set to *fanout*, only max_fanout violations are fixed in final DRC fixing stage of compile command.

When set to *capacitance*, only max_capacitance violations are fixed in final DRC fixing stage of compile command.

If the variable is set to *all*, compile tries to fix all three - max_transition, max_fanout and max_capacitance violations in final DRC fixing stage of compile command.

To determine the current value of this variable, use the *printvar compile_final_drc_fix* command. For a list of all compile variables and their current values, use the *print_variable_group compile* command.

See Also

- compile

---

compile_fix_cell_degradation

Controls whether the algorithms for fixing cell degradation violation are activated.

Data Types

Boolean

Default false in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

When true, the algorithms for fixing cell degradation violations in *compile* and *reoptimize_design* are activated. Different strategies, such as sizing and buffering, try to fix violations of the cell degradation design rule.

To determine the current value of this variable, type *printvar compile_fix_cell_degradation*. For a list of all compile variables and their current values, type *print_variable_group compile*.

See Also

- compile

Feedback

compile_high_effort_area

If this is true, high effort area optimization will be done as part of compile and incremental compile(s). This feature is supported in DCNXT only.

Data Types

Boolean

Default false

Description

The variable *compile_high_effort_area* is used to enable high effort for area optimization in the subsequent compile command(s). Please note that when this variable is enabled, the following variables get enabled by default unless explicilty disabled by the user - *compile_optimize_netlist_area, compile_high_effort_area_in_incremental and compile_optimize_netlist_area_in_incremental This feature is supported in DCNXT only.*

Examples

The following examples show how to enable and disable high effort area optimization during compile:

```
prompt> set compile_high_effort_area true
prompt> set compile_high_effort_area false
```

See Also

- compile_high_effort_area_in_incremental
- compile_optimize_netlist_area
- compile_optimize_netlist_area_in_incremental

compile_high_effort_area_in_incremental

If this is true, high effort area optimization will be done as part of incremental compile(s). This feature is supported in DCNXT only.

Data Types

Boolean

Default default

Description

The *compile_high_effort_area_in_incremental* variable is used to set the area optimization effort as high for the incremental compile command(s). Please note that the default value

of this variable will be set according to the settings used for primary compile. This means, this variable will be set to true by default, if *compile_high_effort_area* is set. Thus if high effort area optimization is enabled/disabled for compile, it will be automatically enabled/ disabled for incremental compile as well. This feature is supported in DCNXT only.

Examples

The following examples show how to explicitly enable and disable high effort area optimization during incremental compile:

```
prompt> set compile_high_effort_area_in_incremental true
prompt> set compile_high_effort_area_in_incremental false
```

See Also

- compile_high_effort_area
- compile_optimize_netlist_area_in_incremental

---

compile_high_effort_area_structuring

Performs high effort area restructuring based optimization by default during compile.

Data Types

Boolean

Default true

Description

The *compile_high_effort_area_structuring* variable performs high-effort area restructuring based optimization by default during compile_ultra. Not available with incremental compile and optimize_netlist -area. Variable helps to improve area in timing constraint context and helps the designs that are constrained for timing. For the designs which are not constrained for timing or under constrained for timing this feature is not required and can be turned off. Feature is available in Topographical SPG flow and WLM flow. Not available in Topographical non-SPG flow. This feature is supported in both Design Compiler and the Design Compiler NXT tools.

Examples

The following example show how to disable high effort area structuring optimization before compile:

```
prompt> set compile_high_effort_area_structuring false
```

Feedback

See Also

- compile_high_effort_area
- compile_high_effort_area_in_incremental
- compile_optimize_netlist_area
- compile_optimize_netlist_area_in_incremental

---

compile_high_pin_density_cell_optimization

Directs the *compile_ultra -spg* command to replace high-pin-density cells with low-pin-density variant-aware electrically equivalent (EEQ) cells that can be routed more easily.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

High-pin-density cells are hard to route in high utilization regions of the design. When you set the *compile_high_pin_density_cell_optimization* variable to *true*, the *compile_ultra -spg* command replaces high-pin-density cells with equivalent low-pin-density variant-aware equivalent EEQ cells to reduce congestion and improve routability with minimal impact to quality of results (QoR).

This feature works only in topographical mode.

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

Examples

The following example enables the variable, and the high-pin-density cells are replaced during synthesis:

```
prompt> set_app_var compile_high_pin_density_cell_optimization true
true
```

Feedback

See Also

- compile_high_pin_density_cell_optimization_utilization_threshold
- compile_ultra
- printvar

---

compile_high_pin_density_cell_optimization_utilization_threshold

Specifies the utilization threshold percentage that determines which regions of the design will be targeted for cell optimization by the *compile_ultra* command.

Data Types

Float. Valid range is from 0.0 to 1.0.

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

0.5 in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

When the *compile_high_pin_density_cell_optimization* variable is set to *true*, the value of the *compile_high_pin_density_cell_optimization_utilization_threshold* variable controls which high-pin-density cells are replaced by electrically equivalent (EEQ) cells by the *compile_ultra -spg* command.

The tool divides the design into smaller areas and computes the regional utilization of these regions. When the utilization of a region is above the user-specified threshold, the high-pin-density cells in these regions are swapped with equivalent low-pin-density electrically equivalent (EEQ) cells that can be routed more easily.

This feature works only in topographical mode.

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

Examples

The following example shows how to set the utilization threshold percentage to identify regions in the design for replacing high-pin-density cells during synthesis:

```
prompt> set
 compile_high_pin_density_cell_optimization_utilization_threshold 0.75
0.75
```

Feedback

See Also

- compile_high_pin_density_cell_optimization
- compile_ultra
- printvar

---

compile_high_pin_density_cell_spreading

Places high pin density cells so they are spread out by the *compile_ultra* command to improve routing.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

High pin density cells are hard to route when they are placed too close together. When you set the *compile_high_pin_density_cell_spreading* variable to *true*, the *compile_ultra* command separates high pin density cells to reduce congestion and improve routability.

This feature works only in topographical mode.

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

Examples

The following example enables the variable and the high pin density cells in the design are spread out during placement:

```
prompt> set_app_var compile_high_pin_density_cell_spreading true
true
```

See Also

- compile_high_pin_density_cell_spreading_threshold
- compile_ultra
- printvar

Feedback

compile_high_pin_density_cell_spreading_threshold

Specifies the pin density threshold percentage that determines which high pin density cells will be spread out by the *compile_ultra* command

Data Types

Float. Valid range is from 0.0 to 1.0.

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

0.9 in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

When the *compile_high_pin_density_cell_spreading* variable is set to *true*, the value of the *compile_high_pin_density_cell_spreading_threshold* variable controls which high pin density cells will be placed with a larger separation by the *compile_ultra* command. The cells with a pin density that exceeds the threshold percentage are spread out during placement,

This feature works only in topographical mode.

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

Examples

The following example shows how to set the pin density threshold percentage to spread out high pin density cells during placement for a design:

```
prompt> set compile_high_pin_density_cell_spreading_threshold 0.75
0.75
```

See Also

- compile_high_pin_density_cell_spreading
- compile_ultra
- printvar

compile_hold_reduce_cell_count

Controls whether the logic used to fix hold time violations is selected based on minimum cell count or minimum area.

Feedback

Data Types

Boolean

Default strict

Description

When true, Design Compiler uses the minimum number of cells to fix the hold time (min path) violations, rather than choosing cells that minimize the total new area. This means that the area may be worsened (compared to the default flow) while the hold time violations are being fixed.

---

compile_identify_synchronous_shift_register_effort

Controls the complexity and the number of synchronous shift registers identified by DC.

Data Types

String

Default none

Description

The variable *compile_identify_synchronous_shift_register_effort* has effect only if the synchronous shift registers identification Is enabled. When the value of this variable is set to *none*, synchronous shift registers (SSR) will be identified during compile_ultra command. However, optimization engines can modify the synchronous logic between the identified synchronous shift registers. Therefore, synchronous shift registers need to be re-identified again in insert_dft. The shift registers identified before insert_dft are not all preserved.

With this effort variable set to *low*, *medium* or *high*, synchronous shift registers identification will happen in insert_dft when most of the optimizations are done. This helps preserving synchronous shift registers during insert_dft.

By setting the variable to *low* , Design Compiler will identify simplest synchronous shift registers. This effort usually provides better area improvements and minimizes the impact on timing. By setting the variable to *medium*, Design Compiler will identify even more synchronous shift registers. This of course produces better sequential area improvement, but timing may be impacted. By setting the variable to *high* , Design Compiler will identify all possible synchronous shift registers. The sequential area is further improved but the impact on timing can be high and overall area might increase.

Feedback

See Also

- compile_seqmap_identify_shift_registers
- compile_seqmap_identify_shift_registers_with_synchronous_logic

---

compile_implementation_selection

Controls whether the *compile* command reevaluates the current implementation of a synthetic module during optimization.

Data Types

Boolean

Default true in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

When this variable is set to true (the default), the *compile* command reevaluates the current implementation of a synthetic library module and replaces it, if appropriate for optimizing the design. When the value is false, this optimization is disabled, saving CPU time with a potential loss of quality of results.

To determine the current value of this variable, use the *printvar compile_implementation_selection* command. For a list of compile variables and their current values, use the *print_variable_group compile* command.

See Also

- compile
- set_implementation

---

compile_inbound_cell_optimization

Enables inbound cell optimization in *compile_ultra*.

Data Types

Boolean

Default false

Feedback

Description

When you set the *compile_inbound_cell_optimization* variable to *true*, the *compile_ultra* command runs a strategy intended to control the usage of inbound cells in the design. This may increase or reduce the number of inbound cells depending on the threshold specified by *compile_inbound_max_cell_percentage*.

Examples

The following example shows how to enable and disable inbound cell optimization for a design:

```
prompt> set compile_inbound_cell_optimization true
prompt> set compile_inbound_cell_optimization false
```

See Also

- compile_ultra
- compile_inbound_max_cell_percentage
- compile_inbound_sitedef_name

---

compile_inbound_max_cell_percentage

Specifies a threshold for the percentage of total inbound cells in the design.

Data Types

float

Default 10.0

Description

The value you specify for the *compile_inbound_max_cell_percentage* variable defines a maximum threshold for the percentage of the total inbound cells in the design. The *inbound cell optimization* will control the number of inbound cells in the design so that the percentage of inbound cells never exceeds the specified value.

Examples

The following example sets the inbound cells threshold to 10%.

prompt> *set compile_inbound_max_cell_percentage 10.0*

Feedback

See Also

- compile_inbound_cell_optimization
- compile_inbound_sitedef_name

---

compile_inbound_sitedef_name

Specifies sitedef name for inbound cells.

Data Types

string

Description

This variable *compile_inbound_sitedef_name* specifies the name of your library's site definition for inbound cells. This name is required for identifying inbound library cells while performing *inbound cell optimization*.

Examples

The following example shows how to set this variable:

```
prompt> set compile_inbound_sitedef_name ibunit
```

See Also

- compile_inbound_cell_optimization
- compile_inbound_max_cell_percentage

---

compile_instance_name_prefix

Specifies the prefix used in generating cell instance names when the *compile* command is run.

Data Types

string

Default U

Description

This variable specifies the prefix used in generating cell instance names when running the *compile* command.

Feedback

To determine the current value of this variable, use the *printvar compile_instance_name_prefix* command. For a list of compile variables and their current values, use the *print_variable_group compile* command.

---

compile_instance_name_suffix

Specifies the suffix used for generating cell instance names when the *compile* command is run.

Data Types

string

Default ""

Description

This variable specifies the suffix used for generating cell instance names when running the *compile* command.

To determine the current value of this variable, use the *printvar compile_instance_name_suffix* command. For a list of *compile* variables and their current values, use the *print_variable_group compile* command.

---

compile_keep_original_for_external_references

Instructs the *compile* command to keep the original design when there is an external reference to the design.

Data Types

Boolean

Default false

Group

none

Description

By default, the *compile* command modifies the original copy of the designs in the current design. When the *compile_keep_original_for_external_references* variable is set to true, the original design and its subdesigns are copied and preserved (before doing any modifications during compile), if there is an external reference to this design.

For example, if there is an instance of a design named *bot, U1* in the current design named *mid*, and there is an external reference from another design named *top* that is not part of current hierarchy, then *U1* will be uniquified to a new design named *bot_0* before

doing any modification to the design. Therefore, when you change the current_design to *top* and perform a link, the original *bot* design will be linked into the current_hierarchy.

Usually this is needed only when you are doing a bottom compile without setting *dont_touch* attributes on all of the subdesigns, particularly with boundary optimization turned on during compile.

If there is a *dont_touch* attribute on any of the instances of the design or in the design itself, this variable does not have any effect.

See Also

- compile

---

compile_log_format

Controls the format of the columns to be displayed during the mapping phases of *compile* and *reoptimize_design* commands.

Data Types

string

Default %elap_time %area %wns %tns %drc %endpoint in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable controls the format of the columns to be displayed during the mapping phases of *compile* and *reoptimize_design* commands. The default specification is shown in the DEFAULT section above and results in an output display format similar to Table 1. The headings and order of the columns displayed correspond to the keywords specified in the syntax. For example, "%elap_time" specifies the ELAPSED TIME column, "%area" the AREA column, and so on.

Table 1

Default Compile Log Output Format

```
                                    TOTAL
ELAPSED                 WORST NEG   SETUP      DESIGN
 TIME        AREA         SLACK      COST     RULE COST       ENDPOINT
--------  ---------  ---------  ---------  ---------
 -------------------------
18:00:30   1498.0       4.12       32.2         0.0
 U1/U2/CURRENT_SECS_reg[4]
18:00:30   1498.0       4.07       31.6         0.0
 U1/U2/CURRENT_SECS_reg[4]
```

```
18:00:30   1497.0      4.07       30.6        0.0
 U1/U2/CURRENT_SECS_reg[4]
18:00:31   1499.0      3.61       27.5        0.0
 U1/U2/CURRENT_SECS_reg[4]
18:00:31   1499.0      3.58       26.1        0.0
 U1/U2/CURRENT_SECS_reg[4]
```

By default, the columns in Table 1 are nine characters wide except for the ENDPOINT column, which is 25 characters. The default precision for the floating-point data types AREA, TOTAL NET SLACK, and DESIGN RULE COST is 1 digit to the right of the decimal point; for WORST NEG SLACK, 2 digits.

There are 13 possible columns that can be displayed; only 6 are displayed in the default format. You can create a customized output format by specifying any number of the available columns, with their keywords and defaults. For example, specifying "%mem" displays the MBYTES column with a width of 6 characters and a precision of 1 digit to the right of the decimal point. When you specify "%mem", the following information is displayed horizontally under these column names: COLUMN HEADER, DATA TYPE, KEYWORD, WIDTH, PRECISION, and FORMAT.

```
MBYTES       floating point     mem         6          1           f
```

See the section "Definitions of Column Fields with Default Values" for descriptions and contents of the fields corresponding to the column headers.

Changing Default Column Parameters

You can change the default column parameters using the optional expression (w.pf,split), as follows:

```
set_app_var compile_log_format = "  %elap_time %area %wns %tns %drc
 %endpoint".
```

where each column entry can be configured as follows:

```
"%keyword(w.pf,split)"
```

The quantities w, p, f, and split are defined as follows:

w

Specifies the column width. Specifying a width less than 6 defaults the width to 6. For string data types, specifying a width greater than 99 defaults the width to 99. For decimal or floating-point data types, specifying a width greater than 25 defaults to 25.

p

For floating-point numbers only. Specifies the precision in number of digits. See also the definition of f.

f

For floating-point numbers only. Specifies the precision format; values are f or g. The value of f specifies that the precision is expressed as the number of digits to the right of the decimal point. The value of g specifies that the precision is expressed as the total number of significant digits. For example, expressing the floating-point number 13.533 with a precision of 3 in the f format reports the number as 13.533 while in the g format the number is reported as 13.5.

split

By default, if the information in a given field exceeds the column width, it pushes out the next field and the next field is not printed on a new line. Specifying split overrides the default and causes the next field to begin on a new line, starting in the correct column.

1998.02 Compile Log Format

The fields for the 1998.02 version are TRIALS, AREA, DELTA DELAY, TOTAL NEG SLACK, and DESIGN RULE COST. The DELTA DELAY field is renamed MAX DELAY COST in the 1998.08 format.

To display the same log format as the 1998.02 version, set the variable as follows:

compile_log_format = ""

Definitions of Column Fields with Default Values

The fields listed under the 5 columns are defined in the following text, showing the default values. Except where noted, units are those defined by the library.

AREA

Shows the area of the design during the optimization.

```
Data type: floating point
Keyword: are
Width: 9
Precision: 1
Format: f
```

CPU SEC

Shows the process CPU time used, in seconds.

```
Data type: decimal
Keyword: cpu
Width: 7
Precision: ignored
Format: ignored
```

DELTA DELAY

See MAX DELAY COST.

DESIGN RULE COST

Measures the distance between the actual results and user-specified design rule constraints.

```
Data type: floating point
Keyword: drc
Width: 9
Precision: 1
Format: f
```

ELAPSED TIME

Tracks the elapsed time since the beginning of the current *compile* or *reoptimize_design*.

```
Data type: string
Keyword: elap_time
Width: 9
Precision: ignored
Format: ignored
```

MAX DELAY COST

Shows the current maximum delay cost of the design, which is the sum of the worst negative slack (max_path violation) in each path group. Called DELTA DELAY in the 1998.02 version.

```
Data type: floating point
Keyword: max_delay
Width: 9
Precision: 2
Format: f
```

MBYTES

Shows the process memory used, in mbytes.

```
Data type: floating point
Keyword: mem
Width: 6
Precision: 1
Format: f
```

MIN DELAY COST

Shows the current minimum delay cost of the design, which is the sum of the worst negative slack (min_path violation) in each path group.

```
Data type: floating point
Keyword: min_delay
Width: 9
Precision: 2
Format: f
```

TIME OF DAY

Shows the current time.

```
Data type: string
Keyword: time
Width: 8
Precision: ignored
Format: ignored
```

TOTAL SETUP COST

Shows the sum of the weighted costs across all endpoints in the design.

```
Data type: floating point
Keyword: tns
Width: 9
Precision: 1
Format: f
```

TRIALS

Tracks the number of transformations that the optimizer tries before making the current selection.

```
Data type: decimal
Keyword: trials
Width: 6
Precision: ignored
Format: ignored
```

WORST NEG SLACK

Shows the worst negative slack (max_path violation) in all path groups.

```
Data type: floating point
Keyword: wns
Width: 9
Precision: 2
Format: f
```

ENDPOINT

Shows the current endpoint being on which work is being done. When the delay violation is being fixed, the object for the ENDPOINT is a cell or a port. When the design rule violations are being fixed, the object for the ENDPOINT is a net.

```
Data type: string
Keyword: endpoint
Width: 25
Precision: ignored
Format: ignored
```

PATH GROUP

Shows the current path group of a valid endpoint.

```
Data type: string
Keyword: group_path
Width: 10
Precision: ignored
Format: ignored
```

DYNAMIC POWER

Shows the dynamic power of the design during optimization.

```
Data type: floating point
Keyword: dynamic_power
Width: 9
Precision: 4
Format: f
```

LEAKAGE POWER

Shows the leakage power of the design during optimization.

```
Data type: floating point
Keyword: leakage_power
Width: 9
Precision: 4
Format: f
```

TOTAL POWER

Shows the total power of the design during optimization. (total_power = dynamic_power + leakage_power)

```
Data type: floating point
Keyword: total_power
Width: 9
Precision: 4
Format: f
```

INSTANCE COUNT

Shows the total cell instance count of the design during optimization.

```
Data type: integer
Keyword: instance_count
Width: 9
```

```
   Precision: ignored
   Format: ignored
```

Examples

The following example increases the precision of the WORST NEG SLACK column by 1 (to 3 digits from its default of 2 digits):

```
prompt> set compile_log_format {%elap_time %area %wns(.3) %tns %drc
 %endpoint}
```

The following example replaces the TOTAL SETUP COST column in the default format with the CPU column:

```
prompt> set compile_log_format {%elap_time %area %wns %cpu %drc
 %endpoint}
```

In the following example, if the ENDPOINT value exceeds the column width, the next field begins on a new line, starting in the correct column:

```
prompt> set compile_log_format {%elap_time %wns %endpoint(19,split)
 %group_path}
```

The following example displays only the MIN DELAY COST column, changes the column width to 12 characters from the default of 9, and expresses the value with a precision of 3 significant digits:

```
prompt> set compile_log_format {%min_delay(12.3g)}
```

The following example sets the compile log to the same format as the 1998.02 version:

```
prompt> set compile_log_format = ""
```

To determine the current value of this variable, use the *printvar compile_log_format* command. For a list of all compile variables and their current values, use *print_variable_group compile*.

See Also

- compile

---

compile_log_print_pathgroups_in_delayopt

Enables printing of Elapsed time, PG, hiers, cells processed in an iteration in delay optimization. This variable is supported in topographical mode in compile_ultra as well as compile_ultra -incremental.This can be used along with the variable "compile_log_format".

Data Types

Boolean

Feedback

Default false

Description

Enables printing of Elapsed time, PG, hiers, cells processed in an iteration in delay optimization. This variable is supported only in topographical mode in compile_ultra as well as compile_ultra -incremental.This can be used along with the variable "compile_log_format".

Allowed values are *false* (the default), *true*.

To determine the current value of this variable, use the *printvar compile_log_print_pathgroups_in_delayopt* command. For a list of all compile variables and their current values, use the *print_variable_group compile* command.

See Also

- compile

compile_mv_no_always_on_buffer_for_port_isolation

If this is true and connected net of the isolating port is always_on, it enable hierarchically down non AO net to be isolated.

Data Types

Boolean

Default false

Description

In case, when isolating port is always_on and the variable is true then instead of isolating the AO net with AO buffer it will isolated hierarchically down non AO net. If there are more than one hierarchically down connected non AO net then port isolation will be skipped.

compile_negative_logic_methodology

Specifies the logic value connected to floating inputs by the *compile* and *translate* commands.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Feedback

Variable is not supported in DC Explorer

Description

When this variable is set to *true*, the *compile* and *translate* commands connect floating inputs to logic 1. When set to the default value of *false*, floating inputs are connected to logic 0.

This variable assignment should be placed at the beginning of the .synopsys_dc.setup file (or dc_shell session) and the value should not be changed during the session.

To determine the current value of this variable, use the *printvar compile_negative_logic_methodology* command. For a list of all compile variables and their current values, use the *print_variable_group compile* command.

---

compile_no_new_cells_at_top_level

Controls whether the *compile* command adds new cells to the top-level design.

Data Types

Boolean

Default false

Description

When this variable is set to *true*, no new cells are added to the top-level design of the hierarchy during *compile*. New cells are added only to lower levels.

This variable is used when the original design has no top-level cells, to prevent the addition of new cells when adding buffers for timing optimization and design rule fixes.

If the design has leaf cells at the top level, the cells are optimized as normal, so this variable should be set to *false* (the default). Setting this variable to *true* means that *compile* will not add any cells during buffering and design rule fixing, even if the design is flat.

Note that if *set_isolate_ports* requires insertion of new cells at the top level, then these cells will be added even if *compile_no_new_cells_at_top_level* is set to *true*. This is because port isolation takes precedence.

To determine the current value of this variable, use the *printvar compile_no_new_cells_at_top_level* command. For a list of all compile variables and their current values, use the *print_variable_group compile* command.

Feedback

compile_optimize_netlist_area

If this is true, area optimization will be done as part of compile and incremental compile(s). This feature is supported in DCNXT only.

Data Types

Boolean

Default false

Description

The *compile_optimize_netlist_area* variable is used to enable area optimization in subsequent compile and incremental compile command(s). This will be enabled by default if *compile_high_effort_area* is turned on. This can be disabled by explicitly resetting the value for this variable. This feature is supported in DCNXT only.

Examples

The following examples show how to enable and disable area optimization during compile:

```
prompt> set compile_optimize_netlist_area true
prompt> set compile_optimize_netlist_area false
```

See Also

- compile_optimize_netlist_area_in_incremental
- compile_high_effort_area

compile_optimize_netlist_area_in_incremental

If this is true, area optimization will be done as part of incremental compile(s). This feature is supported in DCNXT only.

Data Types

Boolean

Default default

Description

The *compile_optimize_netlist_area_in_incremental* variable is used to enable area optimization for the incremental compile command(s). Please note that the default value for this variable will be picked up based on the settings used for primary compile. That means, this variable will be set to true by default, if *compile_optimize_netlist_area* is set. Thus if area optimization is enabled/disabled for compile, it will be automatically enabled/disabled for incremental compile as well. This feature is supported in DCNXT only.

Feedback

Examples

The following examples show how to explicilty enable and disable area optimization during incremental compile:

```
prompt> set compile_optimize_netlist_area_in_incremental true
prompt> set compile_optimize_netlist_area_in_incremental false
```

See Also

- compile_optimize_netlist_area
- compile_high_effort_area_in_incremental

---

compile_optimize_unloaded_seq_logic_with_no_bound_opt

Controls whether unused sequential logic is optimized away when boundary optimization is disabled.

Data Types

Boolean

Default false

Description

When this variable is set to *true*, the tool optimizes away unused sequential logic even when boundary optimization is disabled by setting the *boundary_optimization* attribute to *false* or using the *-no_boundary_optimization* option with the *compile_ultra* command.

In an attempt to remove sequential logic, the tool removes unloaded combinational logic and unloaded nets as well.

If unloaded hierarchical pins are used as connection points for post-compile steps, e.g. clock-gating, dft insertion, etc. then the logic and nets fanning into these pins will need to be protected by a *dont_touch* in order to preserve them.

When the variable is set to *false* (the default), the tool does not optimize unused sequential logic.

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

prompt> *printvar compile_optimize_unloaded_seq_logic_with_no_bound_opt*

Feedback

See Also

- compile_ultra
- set_boundary_optimization
- set_compile_directives

---

compile_power_domain_boundary_optimization

Disables boundary optimization across power domain boundaries when set to false.

Data Types

Boolean

Default true

Group

mv

Description

This variable controls whether to allow boundary optimization across all power domain boundaries. By default, boundary optimization across power domain boundaries is enabled in *compile_ultra*, unless specifically disabled by a command such as *set_boundary_optimization*. This variable provides an automatic way to disable boundary optimization across all power domain boundaries.

See Also

- set_boundary_optimization

---

compile_prefer_faster_flops

Enables compile to prefer faster flops while mapping sequential cells.

Data Types

Boolean

Default false

Description

During mapping by default tool prefers flops with minimum area. Setting this variable to *true*, during mapping compile prefers flops with better delay. This may have area impact, but helps with better timing.

To determine the current value of this variable, use the *printvar compile_prefer_faster_flops* command. For a list of all compile variables and their current values, use the *print_variable_group compile* command.

---

compile_prefer_mux

Maps multiplexing logic in the RTL to MUX trees where necessary to reduce design congestion. This feature is only available in Design Compiler Graphical.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

When you set the *compile_prefer_mux* variable to *true*, the *compile_ultra* command runs a strategy intended to improve the congestion of designs that contain multiplexing logic, possibly at the cost of additional runtime and some area. The default flow typically maps most multiplexers to and-or-invert (AOI) logic in order to minimize area, but in some cases this can result in congestion hotspots. With *compile_prefer_mux* enabled, multiplexing logic that is likely to cause congestion is converted to MUX trees where possible.

The *compile_prefer_mux* variable is supported only in the initial compile step, not in incremental compile.

You can use the variable as an alternative to the *map_to_mux* attribute, which requires you to manually investigate where the attribute needs to be applied.

Examples

The following example shows how to enable and disable congestion-driven MUX optimization for a design. Enable the *compile_prefer_mux* variable by setting it to *true* before running the initial compile.

```
prompt> set compile_prefer_mux true
prompt> set compile_prefer_mux false
```

See Also

- compile_ultra
- set_attribute

Feedback

- get_attribute
- remove_attribute

---

compile_preserve_pin_names_with_sizing

Controls sizing operations to only cells that are pin-name compatible.

Data Types

Boolean

Default false

Description

This option controls how the set of cells that can be used during sizing optimization is determined. It takes effect only on size_only cells. Possible values are *true* and *false*. The default is set to *false*.

For example, assume that there are two buffer cells in the library. One buffer (buffA) has pins "A" and "O" and the other (buffX) has pins "X" and "Z".

When *compile_preserve_pin_names_with_sizing* variable is set to *false*, buffA can be replaced with buffX (and vice versa) during sizing optimization. However, when *compile_preserve_pin_names_with_sizing* is set to true, then buffA and buffX are not interchangeable anymore.

This variable can also cause QoR degradation as the valid set of cells available for sizing will be limited.

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

To set this variable, use *set_app_var compile_preserve_pin_names_with_sizing true*

See Also

- compile_ultra
- set_size_only
- size_cell
- report_size_only

---

compile_preserve_subdesign_interfaces

Controls whether the *compile* command preserves the subdesign interface.

Data Types

Boolean

Default false

Description

When this variable is set to *true*, it disables customization of logic external to a subdesign during *compile*, and preserves the subdesign interface. When set to *false* (the default), *compile* customizes the logic external to a subdesign based on the subdesign's internal logic.

compile_ultra -no_boundary_optimization command sets this variable to *true*.

Optimization across hierarchies can take place in 2 different ways:

(1) Optimizing the logic external to a subdesign, based on that specific subdesign

In this type of optimization, the external logic is customized based on the contents of the subdesign. For example, if one of the output ports of the subdesign is internally grounded, then that constant can be pushed out to the higher level of the design and used at that level to further optimize the design. With this optimization, top-level ("pins-out") functionality of all blocks is preserved. However, if the specification of the lower-level subdesign is then changed; for example, so that the output port is instead tied to a logic 1, the higher level of the design cannot be updated accordingly. The logic propagation already occurred based on the original implementation of the subdesign.

By default, *compile* performs this type of optimization. Setting *compile_preserve_subdesign_interfaces* to *true* disables the optimization.

(2) Optimizing the logic internal to a subdesign, based on its specific instantiation

In Design Compiler, hierarchical optimization of a subdesign's internal logic is referred to as boundary optimization. By default, *compile* does not perform boundary optimization. You can enable boundary optimization for a particular module or for the entire design by running the *compile -boundary_optimization* command or by setting the *boundary_optimization* attribute on the desired object using the *set_boundary_optimization* command.

As a result of boundary optimization, the subdesign is customized for its specific environment. Boundary optimization results in an improved overall cost, but the subdesign is no longer functionally equivalent to its original specification. Therefore, the subdesign cannot be reused in an environment that does not match the environment for which it was optimized. For example, if one of the subdesign's input pins is tied externally to a logic 0, boundary optimization can push that logic constant into the subdesign and further optimize the logic in the subdesign. However, if this subdesign is then used in any environment where the input is no longer tied to a logic 0, the resulting design can be logically incorrect.

Feedback

The *compile_preserve_subdesign_interfaces* variable has no effect on objects that are enabled for boundary optimization.

By default, compile performs (1) described above, but not (2). When boundary optimization is enabled, both types of hierarchical optimization are performed.

Set the *compile_preserve_subdesign_interfaces* variable to *true* whenever the internal functionality of a subdesign might change in the future. In this case, *compile* should be disabled from customizing the external logic based on the internal logic of the block. When the internal logic of all blocks has stabilized, you can set this variable to *false* to allow further optimization of external logic based on the internal logic of the submodules.

To determine the current value of this variable, use the *printvar compile_preserve_subdesign_interfaces* command. For a list of compile variables and their current values, use the *print_variable_group compile* command.

See Also

- compile
- compile_ultra
- set_boundary_optimization
- OPT-113

---

compile_print_crossprobe_info_with_seqmap_messages

Adds cross probing information to register removal seqmap log file messages

Data Types

Boolean

Default false

Description

When this variable is set to *true* before compile, the tool prints cross-probe information (origin RTL file name with its full path and line number) of the registers optimized away with sequential optimization Information messages OPT-1206 (constant registers removed), OPT-1207(unloaded registers removed) and OPT-1215 (registers merged).

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

```
prompt> printvar compile_print_crossprobe_info_with_seqmap_messages
```

Feedback

Examples

Constant register removed:

Information: The register 'a/b/const_0_reg { {/users/testcases/top.v:49} }' is a constant and will be removed. (OPT-1206)

Unloaded register removed:

Information: The register 'a/b/unloaded_reg{ {/users/testcases/top.v:193} }' will be removed. (OPT-1207)

Register merged:

Information: In design 'RAM_0', the register 'mul_reg[1] { {/user/testcases/top.v:113} }' is removed because it is merged to 'mul_reg[0]'. (OPT-1215)

---

compile_register_replication

Controls register replication with the *compile_ultra* and *compile* commands.

Data Types

Boolean

Default default

Description

When this variable is set to *true*, which is the default value when you use the *-timing_high_effort_script* and *-spg* options, the *compile_ultra* command tries to identify registers in the current design that can be split to balance the loads for better QoR. This feature works only in topographical mode.

Register splitting can also be controlled on a design-by-design or cell-by-cell basis using the *set_register_replication* command with the *compile* command. The variable and command do not affect each other and should be controlled independently.

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

For a list of all compile variables and their current values, use *print_variable_group compile*.

See Also

- set_register_replication

Feedback

compile_register_replication_across_hierarchy

Controls whether the *compile* command tries to perform register replication across hierarchies to improve timing.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

When *compile_register_replication_across_hierarchy* is set to *true*, the *compile* command enables register replication. In addition, it creates new ports on instances of subdesigns while performing register replication if it is necessary to improve the timing of the design. Register replication across hierarchies might change the interfaces among subdesign instances. If the interfaces are required to be preserved, this variable should be set to *false* (the default).

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

For a list of all compile variables and their current values, use *print_variable_group compile*.

See Also

- set_register_replication

compile_register_replication_do_size_only

Controls register replication for size-only cells during compile.

Data Types

Boolean

Default true

Description

When this variable is set to *true* (the default), and the *compile_register_replication* variable is also set to *true*, the *compile_ultra* command tries to identify registers in the current

design that can be split to balance the loads for better QoR, even if they are size-only cells. This feature is available only in topographical mode.

To disable register replication for registers with a *size_only* attribute, set the *compile_register_replication_do_size_only* variable to *false*.

You can also control register splitting on a design-by-design or cell-by-cell basis by using the *set_register_replication* command with either the *compile* or *compile_ultra* command. The *compile_register_replication_do_size_only* variable and the *set_register_replication* command do not affect each other and should be controlled independently.

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

For a list of all compile variables and their current values, use *print_variable_group compile*.

See Also

- set_register_replication
- compile_register_replication

---

compile_report_on_unmapped_cells

Controls printing error messages when there are unmapped cells, PVT mismatches or placer errors in the design being synthesized.

Data Types

Boolean

Default false

Description

When this variable is set to *true*, *compile_ultra* will print out OPT-999 error messsage when any of the following condition is met,

- there are unmapped cells in the design

- there are PVT mismatches in the design

- placer has aborted due to over utilization

- basic gates are not available for synthesis

Feedback

The unmapped cells in the design could be combination cells or power management cells. In the case of power management cells, user can refer to,

- TRANS-11 messages to get more details on unampped isolation or enable level shifter cells. TRANS-11 will give details on why an isolation or enable level shifter is left unmapped.

- PWR-662 messages to know the sequential elements with retention constraint that could not be mapped.

For PVT mismatches in the design, user can refer to LIBSETUP-051, LIBSETUP-052, LIBSETUP-053 and LIBSETUP-054 messages to identify the cause of PVT mismatch and fix the design constraints or make suitable PVT matching technology library cells available.

In the event the design has more than one issues listed above, OPT-999 error message will concatenate all the reasons and emit one error message. Here is a sample OPT-999 error message,

Error: compile_ultra is not successful, as there are unmapped isolation and mismatching PVT cells. Refer to TRANS-11, LIBSETUP-051/2/3/4 warning messages. (OPT-999)

In the event there are not any of the issues listed above, *compile_ultra*, will emit OPT-799 information message indicating synthesis is successful. This is how OPT-799 will be,

Information: Design Compiler (DC) completed successfully. (OPT-799)

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

---

compile_restructure_sync_set_reset

Controls whether the *compile_ultra* command tries to perform additional optimizations on the input logic to registers affected by the sync_set_reset attribute.

Data Types

Boolean

Default false

Description

When you set this variable to *true*, the *compile_ultra* command tries to perform additional logic optimization on the driving logic to registers affected by the sync_set_reset attribute.

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

For a list of all compile variables and their current values, use *print_variable_group compile*.

Feedback

compile_retime_exception_registers

Controls whether registers with common path exceptions, including max_path, min_path, multicycle_path, false_path, and group_path, can be moved by adaptive retiming.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable controls whether registers with common path exceptions, including max_path, min_path, multicycle_path, false_path, and group_path, can be moved by adaptive retiming.

For example, if an SDC script contains the following command where *reg* is a register, there is a path exception on *reg*:

```
set_max_delay 1.0 -to reg
```

The variable only affects flows that use the *compile_ultra* command with the *-retime* option.

Allowed values are *false* (the default) and *true*.

If *false* is specified, adaptive retiming does not attempt to move registers with exceptions to improve timing or area.

If *true* is specified, adaptive retiming may try to move registers with the max_path, min_path, multicycle_path, false_path, and group_path path exceptions to improve timing or area.

See Also

- compile_ultra

compile_retime_license_behavior

Controls how the *compile* command behaves when the *optimize_registers* or *balance_registers* attribute is set on a design or parts of a design and the required license(s) (BOA-BRT or DC-Expert) are not available immediately.

Feedback

Data Types

string

Default wait in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable controls how the *compile* command behaves when the *optimize_registers* or *balance_registers* attribute is set on a design or parts of a design and the required licenses (BOA-BRT or DC-Expert) are not immediately available.

Allowed values are *wait* (the default), *stop*, and *next*.

- If *wait* is selected, *compile* waits and checks every 5 minutes until the license becomes available. Once the license is available *compile* resumes running.
- If *stop* is selected, *compile* is aborted immediately when the required license is not available.
- If *next* is selected, retiming is executed with the a license allowing less capabilities. If the *optimize_registers* attribute is set and the BOA-BRT license is not available, retiming will be run with the DC-Expert license. QoR will likely be worse. For the *balance_registers* attribute, there is no lower-level capability, so *next* has the same effect as *stop* for *balance_registers*.

To avoid any waiting or aborting of the *compile* command because of retiming licensing, obtain the required license in dc_shell before the start of *compile*. Use *get_license BOA-BRT* or *get_license DC-Expert* repeatedly until the command is successful.

To determine the current value of this variable, use the *printvar compile_retime_license_behavior* command. For a list of all compile variables and their current values, use *print_variable_group compile*.

See Also

- set_balance_registers
- set_optimize_registers

---

compile_seqmap_disable_qn_pin_connections

Controls whether the *compile* command allows sequential elements to use qn pins for connections.

Feedback

Data Types

Boolean

Default false

Description

When the value of this variable is *true*, the *compile* command will avoid using qn pins when mapping the sequential elements in the design .

To determine the current value of this variable, use the *printvar compile_seqmap_disable_qn_pin_connections* command. For a list of all compile variables and their current values, use *print_variable_group compile*.

See Also

- compile_ultra

---

compile_seqmap_enable_dont_touch_seqcell_mapping

Controls whether the *compile* command allows mapping of sequential cell with dont_touch attribute.

Data Types

Boolean

Default false

Description

When the value is *false* (default value), *compile* will not map the sequential cells with dont_touch attribute.

When this variable is set to *true*, *compile* will map the sequential cells with dont_touch attribute.

To determine the current value of this variable, use the *printvar compile_seqmap_enable_dont_touch_seqcell_mapping* command. For a list of all compile variables and their current values, use the *print_variable_group compile* command.

See Also

- set_dont_touch
- remove_attribute

Feedback

compile_seqmap_enable_output_inversion

Controls whether the *compile* command allows sequential elements to have their output phase inverted. This variable has no effect on the *compile_ultra* command.

Data Types

Boolean

Default false

Description

When the value of this variable is *true*, the *compile* command allows the mapping of the sequential elements in the design to library cells whose output phase is inverted. This can help improve QoR. It is also useful when mapping sequential cells to a target library whose sequential cells have only one type of asynchronous inputs (either set or reset). In this case, the only way to match a sequential cell that uses the missing asynchronous input is to use a library cell with the other type of asynchronous input and to invert the output of that cell.

This variable has no effect on the *compile_ultra* command. To control sequential output inversion for *compile_ultra*, use the *-no_seq_output_inversion* option to *compile_ultra*. Use of the *-exact_map* option would disable sequential output inversion even the value of this variable is true.

Note that when using sequential output inversion, you must use the SVF file to verify the functionality of the design using Formality.

To determine the current value of this variable, use the *printvar compile_seqmap_enable_output_inversion* command. For a list of all compile variables and their current values, use *print_variable_group compile*.

See Also

- compile_ultra

compile_seqmap_honor_sync_set_reset

Controls whether the tool tries to keep the synchronous set and reset logic close to the registers during compile.

Data Types

Boolean

Default false

Feedback

Description

When the value is *true*, flip-flops with synchronous reset or preset pins are mapped in one of the following ways: If your library has registers with synchronous reset (or preset) pins, the reset (or preset) net is connected to the reset (or preset) pin of a register with a dedicated reset (or preset) pin. If your library does not have any registers with synchronous reset (or preset) pins, the tool adds extra logic to the data input to generate the reset (or preset) condition on a register without a reset (or preset) pin. In these cases, Design Compiler attempts to map the logic as close as possible to the data pin to minimize X-propagation problems that lead to synthesis/simulation mismatches. It is still important to use the *sync_set_reset* directive in your RTL so that Design Compiler can distinguish the reset (or preset) signals from other data signals and connect the reset signal as close to the register as possible.

To determine the current value of this variable, use the *printvar compile_seqmap_honor_sync_set_reset* command. For a list of all compile variables and their current values, use the *print_variable_group compile* command.

See Also

- hdlin_ff_always_sync_set_reset

---

compile_seqmap_identify_shift_registers

Controls the identification of shift registers in *compile -scan*. This feature is only supported in test-ready compile with Design Compiler Ultra with a multiplexed scan style.

Data Types

Boolean

Default true

Description

When the value of this variable is set to the default value of *true*, Design Compiler Ultra automatically identifies shift registers in the design during test-ready compile.

When all of the shift registers are identified, only the first register is mapped to a scan cell, while the remaining registers are mapped to non-scan cells. This can save a significant amount of area for designs containing many identified shift registers.

Once these shift registers are identified by Design Compiler Ultra, DFT Compiler will also recognize the identified shift registers as shift-register scan segments. But DFT Compiler will break these scan segments, if necessary, to respect test setup requirements such as maximum chain length.

Feedback

Shift registers that contain synchronous logic between the registers can also be identified if the synchronous logic can be controlled such that the data can be shifted from the output of the first register to the input of the next register. This synchronous logic can either be internal to the register (for example, synchronous reset and enable) or it can be external synchronous logic (for example, multiplexor logic between the registers). For shift registers identified with synchronous logic between the registers, DFT Compiler will add additional logic to the scan-enable signal during scan insertion in order to allow the data to be shifted between the registers when in scan mode. This capability is controlled by the *compile_seqmap_identify_shift_registers_with_synchronous_logic* variable, and is enabled by default. See the *compile_seqmap_identify_shift_registers_with_synchronous_logic* variable man page for details.

Shift-register identification is only supported in test-ready compile with Design Compiler Ultra with a multiplexed scan-style.

Set the *compile_seqmap_identify_shift_registers* variable to *false* if you do not want *compile_ultra -scan* to identify shift registers, or if you want to rescan the shift registers already identified in the design back to scan cells.

The *compile_seqmap_identify_shift_registers_with_synchronous_logic* variable does not have any effect when shift-register identification is disabled with the *compile_seqmap_identify_shift_registers* variable.

See Also

- compile_seqmap_identify_shift_registers_with_synchronous_logic

---

compile_seqmap_identify_shift_registers_with_synchronous_logic

Controls whether shift registers that contain synchronous logic between the registers are identified. This variable only has an effect on DC Ultra optimization when shift register identification is enabled with the *compile_seqmap_identify_shift_registers* variable.

Data Types

Boolean

Default false

Description

When the value of this variable is set to *true* and the value of the *compile_seqmap_identify_shift_registers* variable is set to *true*, DC Ultra automatically identifies shift registers that contain synchronous logic between the registers during test-ready compilation if the synchronous logic can be controlled such that data can be shifted from the output of the first register to the input of the next register. This synchronous logic

can either be internal to the register (for example, synchronous reset and enable), or it can be external synchronous logic (for example, multiplexer logic between the registers).

For shift registers that have been identified, only the first register is mapped to a scan cell while the remaining registers are mapped to nonscan cells. This can save a significant amount of area for designs containing many identified shift registers. This capability is only available with test-ready compilation when using a multiplexed scan style.

After the shift registers are identified by DC Ultra, DFT Compiler also recognizes the identified shift registers as shift-register scan segments. DFT Compiler breaks these scan segments, if necessary, to respect test setup requirements such as maximum chain length. For shift registers identified with synchronous logic between the registers, DFT Compiler adds additional logic to the scan-enable signal during scan insertion to allow the data to be shifted between the registers. This extra logic results in shared paths between the scan-enable signal and the functional logic, so it is important not to set a *dont_touch_network* attribute on the scan-enable ports or signals. A *dont_touch_network* attribute on the scan-enable signal would propagate into functional logic paths, preventing the optimization of those paths and possibly leading to QoR degradation.

To disable timing optimization, use the *set_case_analysis* command on scan-enable ports. To disable DRC fixing, use the *set_ideal_network* command on the scan-enable ports. If the *dont_touch_network* attribute must be used, use the *set_dont_touch_network -no_propagate* command instead to avoid propagation of dont_touch into functional logic.

Set this variable to *false* if you do not want DC Ultra to identify shift registers containing synchronous logic between the registers or if your design flow does not permit the insertion of additional logic on the scan-enable signal.

If you are using an ASCII flow before running DFT Compiler and shift registers with synchronous logic between the registers have been identified in the design, you must set the *compile_seqmap_identify_shift_registers_with_synchronous_logic_ascii* variable to *true* and use the *set_scan_state test_ready* command to infer the shift registers again after reading in the ASCII netlist.

See Also

- set_case_analysis
- set_dont_touch_network
- set_ideal_network
- compile_seqmap_identify_shift_registers
- compile_seqmap_identify_shift_registers_with_synchronous_logic_ascii

Feedback

compile_seqmap_identify_shift_registers_with_synchronous_logic_ascii

Controls the identification of synchronous-logic shift registers by the *set_scan_state test_ready* command in an ASCII netlist flow.

Data Types

Boolean

Default false

Description

When this variable is set to *true*, the *set_scan_state test_ready* command supports re-identification of synchronous-logic shift registers for ASCII netlists. This flow requires a DC-Ultra license and a DFT-Compiler license. In addition, the following related variables must also be set:

```
set_app_var compile_seqmap_identify_shift_registers true
set_app_var
 compile_seqmap_identify_shift_registers_with_synchronous_logic true
```

In a binary (regular) flow, shift register identification is performed by the *compile_ultra -scan* command. The identified shift registers are annotated with attributes that are used by the DFT architect for reporting and scan stitching.

In an ASCII flow, these shift register attributes are not available in the test-ready netlist. As a result, the *set_scan_state test_ready* command will re-identify shift registers when shift register identification is enabled.

When synchronous-logic shift register re-identification is enabled by this variable, the *set_scan_state test_ready* command calls the DC Ultra shift register identification code (which identifies both simple and synchronous-logic shift registers). If this code identifies shift registers differently than the original test-ready synthesis run, the tool restructures the registers, scanning or unscanning registers as needed:

```
prompt> set_scan_state test_ready
Information: Performing full identification of complex shift registers.
 (TEST-1190)
```

This restructuring can improve the quality of results, especially when importing netlists from flows without shift register identification. Although restructuring occurs only when this variable is set, the re-identification processes both simple and synchronous-logic shift registers.

Once the design is in a *test_ready* state, running the *set_scan_state test_ready* command again does not have any effect.

Feedback

If the DC-Ultra and DFT-Compiler licenses are not available, only simple shift registers are identified by the *set_scan_state test_ready* command.

See Also

- compile_seqmap_identify_shift_registers
- compile_seqmap_identify_shift_registers_with_synchronous_logic

---

compile_seqmap_propagate_constant_clocks

Controls whether the *compile* command will propagate constant clock while identifying and removing constant registers.

Data Types

Boolean

Default false

Description

When the value is *false* (default value), *compile* will treat constant clock as normal clock and registers would not be removed due to consatnt clock.

When this variable is set to *true*, *compile* will propagate constant clock so that register with constant clock may be removed as constant register.

To determine the current value of this variable, use the *printvar compile_seqmap_propagate_constant_clocks* command. For a list of all compile variables and their current values, use the *print_variable_group compile* command.

See Also

- compile_seqmap_propagate_constants
- compile_seqmap_propagate_high_effort

---

compile_seqmap_propagate_constants

Controls whether the *compile* command tries to identify and remove constant registers and propagate the constant value throughout the design.

Data Types

Boolean

Default true

Feedback

Description

When the value is *true* (the default), *compile* tries to identify and remove constant sequential elements in the design, which improves the area of the design.

When a constant register is removed, an OPT-1206 message is printed. A register is constant if it is forced into a state (0 or 1) from which it can never escape. A register is also constant if it cannot escape its reset state. The latter behavior is further controlled by the *compile_seqmap_propagate_high_effort* variable.

The reset state of a register is determined by the presence of set and/or reset inputs on that register. For example, if a register has a non-constant reset input and no set input it is assumed that the reset state of the register is logic 0.

To determine the current value of this variable, use the *printvar compile_seqmap_propagate_constants* command. For a list of all compile variables and their current values, use the *print_variable_group compile* command.

See Also

- compile_seqmap_propagate_high_effort

---

compile_seqmap_propagate_constants_size_only

Controls whether the *compile_ultra* command will propagate constant values through registers that are marked size_only when trying to identify and remove constant registers.

Data Types

Boolean

Default false

Description

When the value is *false* (default value), *compile_ultra* will assume that all size_only registers are nonconstant.

When this variable is set to *true*, *compile_ultra* will propagate constant values through size_only registers. It will not remove the size_only registers, however. Even if it finds them to be constant.

To determine the current value of this variable, use the *printvar compile_seqmap_propagate_constants_size_only* command. For a list of all compile variables and their current values, use the *print_variable_group compile* command.

Feedback

See Also

- compile_seqmap_propagate_constants
- compile_seqmap_propagate_high_effort
- set_size_only

---

compile_seqmap_propagate_high_effort

Controls whether the *compile* command considers registers that cannot escape their reset state to be constant.

Data Types

Boolean

Default true in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

When *compile_seqmap_propagate_high_effort* is *true*, the *compile* command tries to identify registers that cannot escape their reset state. Such registers are considered constant and are removed from the design, which improves the area of the design.

The reset state of a register is determined by the presence of set and/or reset inputs on that register. For example, if a register has a non-constant reset input and no set input, it is assumed that the reset state of the register is logic 0. Furthermore, if once this register is in the logic 0 state, it cannot enter the logic 1 state, it is considered to be trapped in the logic zero state and will be removed by *compile* as a constant register.

The *compile* command prints an OPT-1206 message whenever a constant register is removed from the design.

To determine the current value of this variable, use the *printvar compile_seqmap_propagate_high_effort* command. For a list of all compile variables and their current values, use the *print_variable_group compile* command.

See Also

- compile_seqmap_propagate_constants

Feedback

compile_seqmap_report_non_scan_mapping

Controls whether to report the reason or constraint on a register if it cannot be mapped to scan cell in scan replacement.

Data Types

Boolean

Default false

Description

When this variable is set to *true*, the *compile_ultra -scan* and *insert_dft* command analyze library and provide message when a register cannot be mapped to scan cell.

An OPT-1250 message is printed when compile command does not have -scan option or there is no valid scan libcell in the library.

An OPT-1251 message is printed when scan_element attribute is false. The scan_element attribute could be set on registers or design.

An OPT-1252 message is printed when the library has limited libcell type to match the register.

An OPT-1253 message is printed when the cell is unscan for shift-register.

An OPT-1254 message is printed when a mapping constraint is set on a cell, tool might not be able to map to scan cells.

An OPT-1255 message is printed when the register type is not available in the library.

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

For a list of all compile variables and their current values, use *print_variable_group compile*.

See Also

- compile_ultra
- insert_dft

compile_shift_register_max_length

Limit the bit length of shift registers identified in *compile -scan*. This variable only has an effect on DC Ultra optimization when shift register identification is enabled with the *compile_seqmap_identify_shift_registers* variable.

Feedback

Data Types

Integer

Default 0

Description

When the value of this variable is set to an integer equal or larger than *2*, the identified shift register segments would be split into small segments.

It is not meaningful to set value 1, since shift registers are at least 2 bit long.

The feature is designed to target very long shift register segment. By splitting long segments, the congestion issue could be eased in the PnR stage.

Shift register identification would try to balance the length of segments within the limitation of the feature. For example, a 100 length segment with limitation of 80 would be splitted to two 50 segments, instead of 80 and 20.

See Also

- compile_seqmap_identify_shift_registers

---

compile_state_reachability_high_effort_merge

High effort for register merging is a new capability for the *compilecommand. It allows to find more equal and opposite registers.*

Data Types

Boolean

Default false

Description

When *compile_state_reachability_high_effort_merge* is *true*, the *compile* command tries to identify equal and opposite registers by running a more aggressive algorithm.

The *compile* command prints an OPT-1215 message whenever a register is removed from the design because it is equal or opposite to another register.

To determine the current value of this variable, use the *printvar compile_state_reachability_high_effort_merge* command. For a list of all compile variables and their current values, use the *print_variable_group compile* command.

See Also

- compile_seqmap_propagate_constants

Feedback

compile_timing_high_effort

Enables high-effort timing optimizations in *compile_ultra*.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

When you set the *compile_timing_high_effort* variable to *true*, the *compile_ultra* command runs a strategy intended to improve the delay of the design, possibly at the cost of additional runtime. Use the *compile_ultra* command without the variable as the default strategy, and enable this variable to fine tune the flow, as needed, on designs that require further timing optimization. This strategy will perform additional runtime intense passes to achieve better timing optimization.

Examples

The following example shows how to enable and disable high-effort timing optimizations for a design:

```
prompt> compile_timing_high_effort true
prompt> compile_timing_high_effort false
```

See Also

- compile_ultra
- set_attribute
- get_attribute
- remove_attribute

compile_timing_high_effort_tns

Enables high-effort optimization to improve total negative slack when you use the *compile_ultra* command.

Data Types

Boolean

Feedback

Default false

Description

When you set the *compile_timing_high_effort_tns* variable to *true*, the *compile_ultra* command runs a strategy intended to improve the total negative slack (TNS) in the design at the cost of additional runtime.

Use the *compile_ultra* command without the variable as the default strategy. Enable the variable by setting the *compile_timing_high_effort* variable set to *true* to fine-tune the flow, as needed, on designs that require further TNS optimization.

Examples

The following example shows how to enable and then disable high-effort total negative slack optimization for a design:

```
prompt> set_app_var compile_timing_high_effort_tns true

prompt> set_app_var compile_timing_high_effort_tns false
```

See Also

- compile_ultra

---

compile_top_all_paths

Controls whether the *compile -top* command fixes all violations in the design, or only those that cross top-level hierarchical boundaries.

Data Types

Boolean

Default false in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

This variable is for use only with the *-top* option of the *compile* command.

When this variable is set to *true*, the *compile -top* command fixes all design rule violations and all timing violations in the design. When *false* (the default), *compile -top* fixes all design rule violations, but only those timing violations that cross top-level hierarchical boundaries.

Feedback

To determine the current value of this variable, use the *printvar compile_top_all_paths* command. For a list of all compile variables and their current values, use the *print_variable_group compile* command.

See Also

- compile

---

compile_ultra_ungroup_dw

Determines whether to unconditionally ungroup DesignWare cells in the *compile_ultra* flow.

Data Types

Boolean

Default true

Description

In the *compile_ultra* flow, by default all DesignWare hierarchies are unconditionally ungrouped in the second pass of compile.

The *compile_ultra_ungroup_dw* variable determines whether the DesignWare cells are auto-ungrouped in the *compile_ultra* flow. If this variable is set to *false*, *compile_ultra* does not unconditionally ungroup the DesignWare cells.

To determine the current value of this variable, use the *printvar compile_ultra_ungroup_dw* command. For a list of all compile variables and their current values, use *print_variable_group compile*.

See Also

- compile_ultra

---

compile_ultra_ungroup_small_hierarchies

Determines whether to automatically ungroup small hierarchies in the *compile_ultra* flow.

Data Types

Boolean

Default true

Feedback

Description

This variable, when set to *false*, turns off the automatic ungrouping of small user design hierarchies at the beginning of the *compile_ultra* flow.

Alternatively, you can achieve the same results by running the *compile_ultra* command with the *-no_autoungroup* option.

Use the *printvar compile_ultra_ungroup_small_hierarchies* command to determine the current value of this variable.

See Also

- compile_ultra

---

compile_update_annotated_delays_during_inplace_opt

Controls whether *compile -in_place* can update annotated delay values in the neighborhood of swapped cells. It has no effect for *reoptimize_design* and *physopt*, which always update annotated delay values.

Data Types

Boolean

Default true in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

When true (the default value), *compile -in_place* is allowed to modify the values of annotated delays on nets connected to the swapped cells and to remove annotated delays on cells connected to the swapped cells.

When false, *compile -in_place* disallows annotated delays to be modified.

This variable is used to run *compile -in_place* with annotated cell and net delays and allows the optimization to update the annotated delays to reflect the timing changes due to swapping cells.

For example, when a cell is swapped for a cell whose input pins have higher pin capacitances, the transition delay on the cells in the fanin increases and invalidates the delays annotated on these cells. In the same manner, swapping a cell will modify the slope component of the delay of cells in the fanout of the swapped cell, thus invalidating the delay annotated on these cells.

Feedback

With this variable set to true, the annotated delays on cells in the fanin and fanout of the swapped cells are removed so that new cell delays will be calculated. It is important to back-annotate net resistances and especially net capacitances to get accurate cell delays.

With this variable set to true, the annotated delays on nets connected to the swapped cells are incrementally modified to take into account the change in load due to the new pin capacitance of the new cells.

Set this variable to false to disallow modification of annotated net delays and removal of annotated cell delays.

To determine the current value of this variable, type *printvar compile_update_annotated_delays_during_inplace_opt*. For a list of all *compile* variables and their current values, type *print_variable_group compile*. For a list of all *links_to_layout* variables and their current values, type *print_variable_group links_to_layout*.

See Also

- compile

---

complete_mixed_mode_extraction

Enables extraction of both routed and unrouted nets with a single command. This is known as mixed-mode extraction.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

true in Design Compiler topographical mode and Design Compiler NXT topographical mode

Group

physopt

Description

By default, the *extract_rc* command skips unrouted or partially routed nets and extracts only completely routed nets. To estimate partially routed or unrouted nets, you must use the *extract_rc* command with the *-estimate* option. Consequently, you must run both *extract_rc* and *extract_rc -estimate* to completely extract a partially routed design.

Feedback

Mixed-mode extraction simplifies this process by enabling simultaneous handling of routed and unrouted nets. By default, this variable is set to *true* enabling the following capabilities:

- The *extract_rc* command performs detailed extraction first, followed by estimation for broken or unrouted nets.
- When you run the *write_parasitics* command after doing mixed-mode extraction, it writes out mixed-mode parasitics.

To disable the mixed-mode extraction capability, set the value of this variable to *false*.

To see the current value of this variable, use the *printvar complete_mixed_mode_extraction* command.

See Also

- extract_rc

---

cts_clock_source_is_exclude_pin

This variable controls the cascaded create-clock behavior. If this variable is set to true, clock tree synthesis (CTS) marks the clock source of a downstream *create_clock* command as an implicit exclude pin.

Data Types

Boolean

Default true

Description

This variable controls the cascaded create-clock behavior. When set to true, it enables CTS to mark the sources of cascaded create_clocks downstream as implicit exclude pins for the purpose of upstream clocks. This behavior is consistent with the timer behavior for cascaded clocks. For example, the following command:

```
prompt> create_clock CLK1
```

has a *create_clock CLK2* defined in its fanouts at pin PIN2. CTS marks PIN2 as an implicit exclude pin for CLK1. PIN2 is still a valid driver for CLK2. This de-couples the two clocks and prevents quality of results (QoR) degradation resulting from the synthesis order of CLK1 and CLK2. If CLK1 is synthesized first, PIN2 still has a huge design rule checking (DRC) violation because CLK2 has not yet been touched by CTS. This affects the results for CLK1 when CTS tries to balance an unsythesized tree with other synthesized trees.

See Also

- create_clock

Feedback

d

db_load_ccs_data

This variable is obsolete. The tool automatically loads the CCS timing information.

dc_allow_rtl_pg

Allows Design Compiler to read RTL containing a limited set of PG (power/ground) nets and pin connections.

Data Types

Boolean

Description

When *true*, RTL read into Design Compiler can contain some PG nets and PG pin connections. These can represent wiring of macros and other special cells but cannot entail a full power netlist.

PG nets must be declared in the RTL as ordinary wires and/or ports. They must not be driven by constants or declared as supply0/supply1.

The RTL PG nets can be connected to leaf cell pins as long as these pins are declared as PG pins in the technology library. This is how a net is inferred as a PG net. In addition, such nets can also be connected to signal pins of leaf cells.

PG connections can only be made to macros, pad cells, and power management cells, such as power switches, level shifters, and isolation cells. Connections can also be made to hierarchical cells; the associated pins and ports are inferred as PG pins and ports. Design Compiler does not permit PG net connections to standard cells.

If RTL containing PG nets has been read, these nets and pin connections, and associated ports, are removed from the logic netlist. They can optionally be included in Verilog output by using the *write_file -format verilog -pg* command. If the *-pg* option is not used, the PG nets and ports will not appear.

In the UPF flow, execution of the UPF commands 'create_supply_port supply_port' and 'create_supply_net supply_net' will result in the logical port/net being removed from the logical netlist. Also, execution of these commands will not throw up any error messages related to name space conflicts.

Feedback

See Also

- dc_allow_rtl_pg_to_analog_pins
- dc_allow_rtl_pg_to_signal_pins

---

dc_allow_rtl_pg_to_analog_pins

Allows an RTL containing PG (power/ground) connections to analog signal pins to be read in successfully.

Data Types

Boolean

Description

When this variable set to *true* (the default), an RTL containing PG connections to analog signal pins will be read in successfully. These connections will be preserved in the Verilog netlist, and also in DDC. A signal pin is identified as an analog signal pin by the presence of "is_analog" Liberty attribute.

When this variable is set to *false*, PG connections to analog signal pins are not allowed. Such connections will result in error messages during link. The connections to analog signal pins will be dropped and the analog signal pins will be connected to constants.

---

dc_allow_rtl_pg_to_signal_pins

Allows an RTL containing PG (power/ground) connections to signal pins to be read in successfully.

Data Types

Boolean

Description

When this variable is set to *true* (the default), an RTL containing PG connections to signal pins will be read in successfully. These connections will be preserved in the Verilog netlist, and also in DDC.

When this variable is set to *false*, PG connections to signal pins are not allowed. Such connections will result in UID-622/UID-632 error messages during link. The connections to signal pins will be dropped and the signal pins will be connected to constants.

```
There are 2 variables now to control RTLPG connections to
analog/signal pins -
1. dc_allow_rtl_pg_to_analog_pins: Allows RTLPG connections to
   analog pins only.
```

Feedback

2. **dc_allow_rtl_pg_to_signal_pins:** Allows RTLPG connections to all signal pins.

```
Here is the table listing the tool behavior in the presence of the
2 variables -
signal_pins | analog_pins | Behavior
-----------------------------------------------------------------
true        |           * | Connections to all signal pins
                            will be allowed. When signal_pins = true,
                            the value of analog_pins will be a NO OP.
false       |        true | Connections to only analog pins
                            will be allowed. Connections to signal pins
                            will be dropped with UID-622/UID-632 error
                            messages and the signal pins will be
                            connected to constants. This will achieve
                            backward compatibility.
false       |       false | Connections to analog/signal pins
                            will be dropped with UID-622/UID-632 error
                            messages and the analog/signal pins will be
                            connected to constants.
```

See Also

- dc_allow_rtl_pg
- dc_allow_rtl_pg_to_analog_pins

---

dc_shell_mode

Reports the mode of the current dc_shell session.

Data Types

string

Default tcl

Description

This variable is set to the mode of the application currently running. The value of this variable can be either *default* or *tcl*. A value of *default* means that the application is running in dcsh default mode. A value of *tcl* means that the application is running in Tcl mode. The variable is read-only.

To determine the current value of this variable, use the *printvar dc_shell_mode* command in Tcl mode, or use the *printvar dc_shell_mode* command in dcsh.

Feedback

dcnxt_adaptive_disable_for_sched_affinity

Disables the use of adaptive multithreading features when manual management of core assignments is detected.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, and DC Explorer

false in Design Compiler NXT non-topographical mode and Design Compiler NXT topographical mode

Description

When execution on more than one core is requested using *set_host_options -max_cores*, by default DCNXT optimizes multithreading performance by dynamically adapting its workload to the available CPU resources. See the manpage for *dcnxt_adaptive_multithreading* for details.

The variable *dcnxt_adaptive_disable_for_sched_affinity* controls how DC-NXT behaves when it detects that scheduling affinity settings are in use for the current process. Such manual management of core assignments cause the machine performance metrics used by DC-NXT to guide multithreading use, to not accurately reflect the actual availability of core resources. Hence, DC-NXT may not be able to correctly adjust its multithreading behavior in that environment.

By default, *dcnxt_adaptive_disable_for_sched_affinity* is set to *true*. Under that setting, DC-NXT disables the adaptive multithreading feature when manual core management is detected.

To continue to use adaptive multithreading when scheduling affinity constraints exist, set the *dcnxt_adaptive_multithreading* variable to *false*.

The default for the *dcnxt_adaptive_disable_for_sched_affinity* variable is *true*.

See Also

- dcnxt_adaptive_multithreading
- set_host_options

dcnxt_adaptive_multithreading

Enables improved multithreading performance by adapting workload to available resources.

Feedback

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, and DC Explorer

true in Design Compiler NXT non-topographical mode and Design Compiler NXT topographical mode

Description

When execution on more than one core is requested using *set_host_options -max_cores*, DCNXT optimizes multithreading performance by dynamically adapting its workload to the available CPU resources. This improves execution time when the machine loads exceeds the number of physical cores. When the machine load is less than the number of physical cores, it does not affect execution time. No user configuration is required.

By default, this capability is enabled in DCNXT. You can disable this feature by setting the *dcnxt_adaptive_multithreading* variable to *true*.

The default for the *dcnxt_adaptive_multithreading* variable is *true*.

This feature is exclusive to DCNXT. It supersedes resource checks controlled by the *disable_multicore_resource_checks* variable, which adjust the number of cores in use based on machine load. When *dcnxt_adaptive_multithreading* is *true*, those checks are disabled.

See Also

- set_host_options
- disable_multicore_resource_checks

---

dcnxt_enable_ndm_flow_for_adv_nodes

Enables converting Milkyway libraries to NDM libraries for advance nodes

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

true in Design Compiler NXT topographical mode

Feedback

Description

When variable *dcnxt_enable_ndm_flow_for_adv_nodes* is set to *true* before invoking *create_mw_lib* command and tool understands the provided libraries and technology file are for advance nodes that is 5nm and below, it automatically triggers a flow to convert given Milkyway libraries to NDM libraries and initializes the *Design Compiler NXT* into NDM mode.

This feature in supported in Design Compiler NXT topographical mode.

Examples

The following example shows how to enable and disable the automatic NDM mode

```
prompt> set dcnxt_enable_ndm_flow_for_adv_nodes true
prompt> set dcnxt_enable_ndm_flow_for_adv_nodes false
```

See Also

- create_mw_lib
- create_lib
- shell_is_in_ndm_mode

---

dcnxt_exclude_physical_hierarchy_macro

Controls the feature to exclude/include the hierarchical macro cell in the create_auto_path_groups command.

Data Types

Boolean

Description

When this variable is set to *false*, hierarchical macro cell is included by the create_auto_path_groups command.

When this variable is set to *true* (the default), hierarchical macro cell is excluded from the path groups.

Below is how the flow looks like in the presence of this variable.

```
>>>>
set dcnxt_exclude_physical_hierarchy_macro false

set_physical_hierarchy I_MIDDLE_0
create_auto_path_groups -mode mapped

>>>>
```

Feedback

See Also

- create_auto_path_groups

---

dcnxt_ndm_library_configuration_mw_exec

Specifies Milkyway executable path to convert Milkyway libraries to NDM libraries.

Data Types

String

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

"" in Design Compiler NXT topographical mode

Description

Commands *create_mw_lib* and *create_lib* are enhanced to accept *Milkyway libraries*, and automatically triggers a flow to convert given *Milkyway libraries* to NDM libraries.

The feature needs one of the IC Compiler or Milkyway executable to read the given Milkyway libraries. Tcl variable *dcnxt_ndm_library_configuration_mw_exec* can be used to specify the Milkyway shell before invoking commands *create_mw_lib* and *create_lib*:

This feature is supported in Design Compiler NXT topographical mode.

Examples

The following example shows how to set output NDM directory

```
prompt> set dcnxt_ndm_library_configuration_mw_exec "/path/Milkyway"
```

See Also

- create_mw_lib
- create_lib
- shell_is_in_ndm_mode

---

dcnxt_skip_writing_sca_for_hier_blocks

Controls the feature to bypass writing user-specified and tool generated set_case_analysis constraint of the abstract block pin in the top design SDC file, when writing it using write_sdc command.

Feedback

Data Types

Boolean

Description

When this variable is set to *false*, "write_sdc" command will write user-specified as well as tool generated set_case_analysis constraint of the abstract block pin in the top design SDC file.

When this variable is set to *true* (the default), no set_case_analysis for abstract block pin will be written in the top design SDC file.

Below is how the flow will look like in the presence of this variable.

```
>>>>
set dcnxt_skip_writing_sca_for_hier_blocks true

write_sdc output.sdc

>>>>
```

See Also

- write_sdc

---

dcnxt_topo_enable_opcond_matching_for_non_mv

When set, enables MV opcond matching for non-MV flows outside ICC2 link based placement.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

false in Design Compiler NXT topographical mode

Description

When variable *dcnxt_topo_enable_opcond_matching_for_non_mv* is set to *true* before *compile_ultra*, the tool invokes MV infrastructure to enable MV linking. Setting this variable to true may cause QoR difference due to MV aware library linking.

This feature in supported in Design Compiler NXT topographical.

Feedback

Examples

The following example shows how to enable and disable MV infrastructure for non-MV designs

```
prompt> set dcnxt_topo_enable_opcond_matching_for_non_mv true
prompt> set dcnxt_topo_enable_opcond_matching_for_non_mv false
```

---

dct_enable_track_auto_fill

Determines if congestion estimation in Design Compiler Topographical should use layer tracks based on the layer pitch.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

true in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

The setting of the *dct_enable_track_auto_fill* variable determines whether to fill empty space with routing tracks and how congestion estimates the number of available tracks in Design Compiler topographical.

When set to *true*, Design Compiler Topographical congestion estimation fills tracks across the core area according to the pitch of the layer. This will emulate evenly distributed tracks on the design.

When set to *false*, Design Compiler Topographical congestion estimation uses actual tracks as defined in the floorplan. In this case, congestion report and congestion map use actual track availability.

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

See Also

- report_congestion

---

dct_enable_va_aware_ao_synthesis

Sets the flag to enable the voltage aware buffering on physical feedthrough nets during compile_ultra in Design Compiler Topographical.

Feedback

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Setting the *dct_enable_va_aware_ao_synthesis* variable to *true* enables voltage aware AO synthesis on certain physical feedthrough paths. It enables buffer insertion in physical feedthrough path when there is disjoint voltage area, and the physical feedthrough net is very long, but logically they belong to same hierarchy.

See Also

- compile_ultra

---

dct_ignore_special_pg_nets

Controls the feature to ignore specialnets of type Power and Ground when reading the DEF file using extract_physical_constraints command.

Data Types

Boolean

Description

When this variable is set to *true*, "extract_physical_constraints" will skip the special nets of type Power and Ground present in the input DEF file.

When this variable is set to *false* (the default), "extract_physical_constraints" will read the special nets of type Power and Ground present in the input DEF file.

Below is how the flow will look like in the presence of this variable.

```
>>>>
set dct_ignore_special_pg_nets true

read_verilog

extract_physical_constraints
>>>>
```

Feedback

See Also

- extract_physical_constraints

---

dct_maskshift_consistency_check

Sets the flag to enable color mask shift consistency checking during extract_physical_constraints in Design Compiler Topographical.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Setting the *dct_maskshift_consistency_check* variable to *true* enables color mask shift consistency checking between the input DEF and the reference libraries during *extract_physical_constraints*.

These checks work only in topographical mode.

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

See Also

- extract_physical_constraints

---

dct_placement_ignore_scan

Sets the flag for the placer to ignore scan connections in Design Compiler topographical.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Feedback

Description

Setting the *dct_placement_ignore_scan* variable to *true* flags the placer to ignore scan connections, particularly for a scan-stitched design. This is similar to the *create_placement* command with the *-ignore_scan* option in IC Compiler.

See Also

- compile_ultra

---

dct_port_dont_snap_onto_tracks

Sets the flag so that ports are not snapped onto tracks after pin placement in Design Compiler in topographical mode.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

If you set the *dct_port_dont_snap_onto_tracks* variable to *true*, the ports are not snapped onto tracks after pin placement. By default, the ports are snapped onto tracks.

See Also

- compile_ultra

---

dct_preserve_all_preroutes

Determines whether all preroutes are preserved during compilation.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Feedback

Description

If you set the *dct_preserve_all_preroutes* variable to *true*, all preroute elements in the design are preserved as they are. By default, the preroutes of non-existent non-P/G nets are filtered out and removed from the database. Preserving all preroute elements during compilation can increase the memory usage during the *compile_ultra* command run.

See Also

- compile_ultra

---

dct_remove_invalid_bounds

Determines whether invalid bounds and voltages areas identified through the PDC (Physical Data Check) tag are removed during compilation.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, and DC Explorer

false in Design Compiler NXT non-topographical mode and Design Compiler NXT topographical mode

Description

If you set the *dct_remove_invalid_bounds* variable to *true*, all invalid bounds and voltages areas identified through the PDC (Physical Data Check) tag are removed from the design during compile_ultra. Correcting invalid bounds is important for the tool to run smoothly. This feature is exclusive to DCNXT.

See Also

- compile_ultra

---

ddc_allow_unknown_packed_commands

Causes the *read_file* command to attempt to read .ddc files that contain packed commands that are unknown to the current version of *dc_shell*.

Data Types

Boolean

Default true

Feedback

Description

When this variable is set to *true* (the default), the *read_file* command attempts to read a .ddc file that contains embedded commands (constraints) that are unknown to the current version of *dc_shell*. This might allow the tool to read a .ddc file that was written by a newer version of *dc_shell* which, for some reason, is not currently available.

If this feature is enabled, any unrecognized commands are discarded, possibly resulting in the loss of important data. It is best that .ddc files be read by the same (or later) version of the tool as was used to write them out.

If the variable is set to *false*, any attempt to read a file with unknown embedded commands results in a read failure and DDC-6 error message. You might want to set this variable to *false* as a check against possible loss of constraint data.

It may not be possible to read certain DDC files written by newer versions of the tool, even with this variable set to *true*.

This variable is only evaluated during the *read_file* command. Unknown commands read in from DDC files while this variable is *true* will continue to be skipped, even if the variable is subsequently set to *false*.

See Also

- read_file
- DDC-6
- DDC-12
- DDC-13

---

de_enable_physical_flow

Allows DC Explorer to perform optimization with physical design data.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

false in DC Explorer

Feedback

Description

By default, the optimization does not include any physical design data. When this variable is set to true, DC Explorer can perform optimization using floorplan information or physical constraints.

---

de_enable_upf_exploration

Enable the UPF exploration mode in DC Explorer.

Data Types

Boolean

Default true

Group

none

Description

This variable determines if the UPF exploration mode is turned on in DC Explorer.

See Also

- insert_mv_cells

---

de_log_html_filename

Changes the file name of the HTML log file.

Data Types

string

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

default.html in DC Explorer

Group

system_variables

Description

Use this variable to change the file name of the compile log in HTML format that is generated in de_shell.

Feedback

de_log_redirect_enable

Enables redirection of warnings and messages in the DC Explorer log file to the HTML file.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

true in DC Explorer

Group

system_variables

Description

By default, uncritical warnings and messages from the DC Explorer log file are redirected to HTML file. When this variable is set to false, the messages and warnings stay visible only in the log file.

de_log_show_derived_ideal_nets

Enables reporting of nets marked as ideal in DE.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

false in DC Explorer

Description

By default, information message about a net marked as ideal in DE is not printed. When this variable is set to true, the messages are printed in the log file.

de_rename_shell_name_to_dc_shell

de_rename_shell_name_to_dc_shell

Feedback

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

false in DC Explorer

Group

none

Description

To change the *synopsys_program_name* variable setting to *dc_shell* during the *de_shell* session, set this variable to true.

```
prompt> echo $synopsys_program_name
de_shell
prompt> set de_rename_shell_name_to_dc_shell true
true
prompt> echo $synopsys_program_name
dc_shell
prompt> set de_rename_shell_name_to_dc_shell false
false
prompt> echo $synopsys_program_name
de_shell
```

To determine the current setting of this variable, use the *get_app_var de_rename_shell_name_to_dc_shell* command.

---

default_input_delay

Specifies the global default input delay value to be used for environment propagation.

Data Types

float

Default 30 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

Specifies the global default input delay value to be used for environment propagation. This variable is used by the *derive_constraints* command.

Feedback

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

See Also

- derive_constraints

---

default_name_rules

Contains the name of a name rule to be used as a default by the *change_names* command, if the command's *-rules* option does not specify a *name_rules* value.

Data Types

string

Default ""

Description

This variable contains the name of a name rule to be used as a default by the *change_names* command, if the command's *-rules* option does not specify a *name_rules* value.

The *change_names* command changes the names of ports, cells, and nets to conform to the rules specified by *default_name_rules*. The *name_rule* you assign to *default_name_rules* must already have been defined using *define_name_rules*. For information on the format and creation of *name_rules*, see the *define_name_rules* man page.

To determine the current value of this variable, use the *printvar default_name_rules* command. For a list of all system variables and their current values, use *print_variable_group system*.

See Also

- change_names
- define_name_rules

---

default_output_delay

Specifies the global default output delay value to be used for environment propagation.

Data Types

float

Feedback

Default 30 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable specifies the global default output delay value to be used for environment propagation. This variable is used by the *derive_constraints* command.

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

See Also

- derive_constraints

---

default_port_connection_class

Contains the value of the connection class to be assigned to ports that do not have a connection class assigned to them.

Data Types

string

Default universal

Description

This variable specifies the value of the connection class to be assigned to ports that do not have a connection class assigned to them. The default value for *default_port_connection_class* is *universal*.

To determine the current value of this variable, use the *printvar default_port_connection_class* command. For a list of all compile variables and their current values, use the *print_variable_group compile* command.

See Also

- set_connection_class

---

default_schematic_options

Specifies options to use when schematics are generated.

Feedback

Data Types

string

Default -size infinite in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable options to use when schematics are generated. When the value is set to *-size infinite* (the default), the schematic for a design is displayed on a single page. This is used by the Design Analyzer.

To determine the current value of this variable, use the *printvar default_schematic_options* command. For a list of all schematic or view variables and their current values, use either the *print_variable_group schematic* or the *print_variable_group view* command.

---

designer

Specifies the name of the current user.

Data Types

string

Default ""

Description

This variable specifies the name of the current user. This name is displayed on the schematics.

To determine the current value of this variable, use the *printvar designer* command. For a list of all system variables and their current values, use the *print_variable_group system* command.

---

dft_enable_mis_cells_decomposition

If this is true, multi-input switching cells will be decomposed as part of dft insertion.

Data Types

Boolean

Default false

Feedback

Description

The variable *dft_enable_mis_cells_decomposition* is used to enable or disable decomposition of MIS cells in the subsequent insert_dft command(s).

Examples

The following examples show how to enable and disable decomposition of MIS cells during insert_dft command:

```
prompt> set dft_enable_mis_cells_decomposition true
prompt> set dft_enable_mis_cells_decomposition false
```

---

dft_tp_enable_mv_checks

Enables power aware test point insertion.

Data Types

string

Default all

Description

To enable the power aware test point insertion. There are five valid values (multiple values are supported):

"location": Test point will not be inserted if it introduces an MV violation.

"clock": Test point will not be inserted if its clock introduces an MV violation.

"control": Test point will not be inserted if its control signal introduces an MV violation.

"all": Test point will not be inserted if its location, clock or control signal introduces an MV violation.

"none": To disable the power aware test point insertion.

Examples

In the following example, the test point will not be inserted if its clock or control signal introdues an MV violation:

```
prompt> set dft_tp_enable_mv_checks "clock control"
```

---

dft_tp_show_driver_mapping

prints driver mappings for test points in testpoint.rpt

Feedback

Data Types

Boolean

Default false

Description

prints driver mappings for test points in testpoint.rpt

Examples

In the following example, the test point driver mapping will be printed if the value is true:

```
prompt> set dft_tp_show_driver_mapping "true"
```

---

disable_auto_time_borrow

Determines whether the *report_timing* command and other commands will use automatic time borrowing.

Data Types

Boolean

Default false

Group

timing_variables

Description

This variable determines whether the *report_timing* command and other commands will use automatic time borrowing. When the value is *false* (the default), automatic time borrowing occurs. Automatic time borrowing balances the slack along back-to-back latch paths, to reduce the overall delay cost. Allocating slack throughout the latch stages can improve optimization results.

When set to *true*, no slack balancing occurs during time borrowing. This means the first paths borrow enough time to meet the constraint until the *max_time_borrow* is reached. Setting the variable to *true* produces time-borrow results consistent with PrimeTime.

See Also

- report_timing

Feedback

disable_case_analysis

Disables constant propagation from both logic constants and *set_case_analysis* command constants when set to *true*.

Data Types

Boolean

Default false

Group

timing

Description

When this variable is set to *true*, it disables constant propagation from both logic constants and *set_case_analysis* command constants. By default, the *disable_case_analysis* variable is *false*, so constant propagation is performed if there has been a *set_case_analysis* command, or if the *case_analysis_with_logic_constants* variable has been specified.

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

See Also

- remove_case_analysis
- report_case_analysis
- set_case_analysis
- case_analysis_with_logic_constants

disable_library_transition_degradation

Controls whether the transition degradation table is used to determine the net transition time.

Data Types

Boolean

Default false

Group

timing_variables

Feedback

Description

When this variable is set to *false* (the default), *report_timing* and other commands that use timing in dc_shell use the transition degradation table in the library to determine the net transition time. When *true*, the timing commands behave as though there were no transition degradation across the net.

See Also

- report_timing

---

disable_mdb_stop_points

Disables the "stop at any level" functionality in the *write_mdb* command.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

By default, the *write_mdb* command traverses the entire hierarchy for the specified design and writes all cell information into the destination Milkyway database. To increase the capacity and to reduce the runtime, you can also specify some modules as "stop point" by setting the *is_mdb_stop_point* to *true*. The *write_mdb* command then handles these modules as leaf modules and does not explore their submodules.

This function can be turned off by setting *disable_mdb_stop_points* to *true*. This setting causes the *write_mdb* command to ignore all stop points.

See Also

- set_attribute

---

disable_multicore_resource_checks

Disables the resource checks on the host machine for the multicore flow.

Data Types

Boolean

Feedback

Default false

Description

This feature is superseded by *dcnxt_adaptive_multithreading*. When that variable is *true*, the checks described here are not performed.

When *dcnxt_adaptive_multithreading* is false, the tool checks resource requirements in the multicore flow, such as the number of cores available and the load average on the host machine. If the host machine does not meet the resources that you specified with the *set_host_options* command, the tool overrides your *set_host_options* settings and uses the optimal resources available on the host machine.

You can override this behavior by setting the *disable_multicore_resource_checks* variable to *true*. In this case, the tool disables the resource checking on the host machine and uses your *set_host_options* settings. However, this can slow down the tool if the host does not have the necessary cores or is heavily loaded.

The default for the *disable_multicore_resource_checks* variable is *false*.

See Also

- dcnxt_adaptive_multithreading
- UIO-230
- UIO-231

---

dont_bind_unused_pins_to_logic_constant

Specify whether timing analysis should apply case analysis of '0' to undriven input pins.

Data Types

Boolean

Default false

Description

This variable controls how the timing engine (i.e., the *update_timing* command) handles undriven leaf and hierarchical input pins.

When this variable is set to its default of *false*, the timing engine applies a case analysis value of 0 at undriven pins. This is consistent with how synthesis optimizes the logic.

When this variable is set to *true*, the timer no longer assumes any case-analysis constant value at undriven input pins.

Feedback

This variable only affects timing analysis. It has no effect on optimization, which always assumes logic 0 at undriven pins.

See Also

- set_case_analysis

---

dont_tie_unused_hierarchical_iopins_to_logic_constant

Specifies if unused hierarchical iopins (no driver and no load) should be connected to constant tie-off cells during loading of the design.

Data Types

Boolean

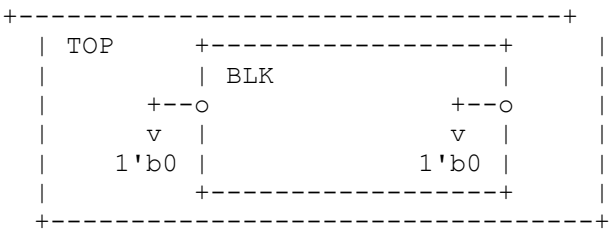
Default false

Description

This variable controls whether unused hierarchical iopins should connect to constant tie-off cells during loading of the design.

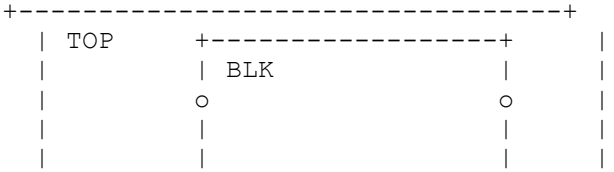
An unused hierarchical iopin is a pin of a hierarchical cell that has no drivers and no loads associated with it. This means that there are no leaf cells and no ports connected to the hierarchical pin. Nets and connections to other hierarchical iopins are not accounted.

When you set this variable to false (the default), the tool ties unused hierarchical iopins to logic 0 during loading of the design, as shown in the following diagram:



For unused input pins, the constant is driven in the parent hierarchy. For unused output pins, the constant is driven from inside the hierarchical block.

When you set this variable to *true* before reading in the design, the tool leaves unused hierarchical iopins undriven, as shown in the following diagram:



Feedback

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

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

See Also

- read

---

dont_touch_nets_with_size_only_cells

Specifies whether a net is marked as dont_touch when it connects at least one leaf-level size-only driver and at least one leaf-level size-only load.

Data Types

Boolean

Default false

Description

When set to *true*, this variable indicates whether nets with one of the following types of connections will have an implicit dont_touch attribute marking. These connections are considered transparently across design hierarchies.

- The net connects a leaf-level size-only driving cell and at least one leaf-level size-only load cell
- The net connects a top-level input port and a leaf-level size-only load cell
- The net connects a leaf-level size-only driving cell and a top-level output port

If a net is explicitly set as *dont_touch false* by using the *set_dont_touch* command, this dont_touch setting supersedes the dont_touch marking derived from the variable.

Cells can be marked as size-only by using the *set_size_only* command.

Be cautious when setting this variable to *true*. If a net is marked as dont_touch, it cannot be changed by commands that optimize or manipulate the design. In particular, dont_touch feedthrough nets and the dont_touch nets connected to two output ports will not be fixed.

The default value of this variable is *false*.

To determine the current value of this variable, use the *printvar dont_touch_nets_with_size_only_cells* command. For a list of compile variables and their current values, use the *print_variable_group compile* command.

Feedback

See Also

- set_size_only
- set_dont_touch
- report_dont_touch

duplicate_ports

Specifies whether ports are to be drawn on every sheet for which an input or output signal appears.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable controls the partitioning option that specifies if ports are to be drawn on every sheet for which an input or output signal appears. When set to *true*, no off-sheet connectors are used for input and output signals, and signal ports are duplicated where indicated on each sheet.

To determine the current value of this variable, use the *printvar duplicate_ports* command. For a list of all schematic variables and their current values, use the *print_variable_group schematic* command.

e

echo_include_commands

Controls whether the contents of a script file are printed as it executes.

Data Types

Boolean

Default true

Feedback

Description

When this variable is set to *true* (the default value), the *include* command prints the contents of files as it executes. When set to *false*, the contents of the files are not printed.

To determine the current value of this variable, use the *printvar echo_include_commands* command. For a list of all system variables and their current values, use the *print_variable_group system* command.

---

enable_bit_blasted_bus_linking

Resolves the mismatches between bit-blasted bus bits and the buses during the link process.

Data Types

Boolean

Default true

Description

Setting this variable to true allows the linker to recognize bit-blasted buses by pin names during the link process. If the pin names follow a specific pattern, the tool infers a bus when encountering the individual bits.

Typically, the RTL pin names and the logic library pin names should match. Signals are defined the same way in both the RTL and the library. They are defined as buses or a bus is defined by its individual wires. Occasionally, mismatches occur during design development; for example, when you transfer the design from one technology to another. To fix the mismatches, you can set the *enable_bit_blasted_bus_linking* variable to true before the design is read.

When you set the *enable_bit_blasted_bus_linking* variable to true, the linker allows you to read your design even if pin name mismatches occur. By default, this variable is set to false, but the default for DC Explorer is true. Note that when this variable is set to true, the tool matches pin names according to the setting of the *bit_blasted_bus_linking_naming_styles* variable.

You specify the pin name patterns by using the *bit_blasted_bus_linking_naming_styles* variable. For example:

```
prompt> set bit_blasted_bus_linking_naming_styles {%s[%d]}
```

This permits the U1 reference

```
input [1:0] in;
my_macro U1( .S(in),...
```

Feedback

to link even if my_macro is defined as follows:

```
my_macro
input S[1];
input S[0];
\...
```

To determine the current value of this variable, use the *printvar enable_bit_blasted_bus_linking* command. For a list of all system variables and their current values, use *print_variable_group system*.

See Also

- bit_blasted_bus_linking_naming_styles

---

enable_cell_based_verilog_reader

Turns on the verilog2cel Verilog reader.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable enables the verilog2cel Verilog reader. which is a CEL-based Verilog reader that saves the netlist directly into the Milkyway CEL, without creating tool data structures.

To determine the current value of this variable, use the *printvar enable_cell_based_verilog_reader* command. For a list of all HDL variables and their current values, use the *print_variable_group hdl* command.

See Also

- read_verilog

---

enable_clock_to_data_analysis

Enables the usage of latency and transition time of ideal clocks for timing analysis in clock-to-data signal paths.

Feedback

Data Types

Boolean

Default false

Description

This variable, when set to *true*, enables timing analysis of clock-to-data pins when the clock is ideal. The ideal clock latency is used as the data arrival time at the clock-to-data pin, and the ideal transition is used as data transition time at that pin. If more than one clock drives the data pin, the worst latency and transition values apply.

When the variable is set to *false* (the default), the tool uses the actual propagated clock latency and transition time at the clock-to-data pin, if any, and ignores any ideal clock latency and transition time set for the clock.

A clock-to-data pin is a cell's data input pin in the fanout of a clock, such as the input of a clock divider circuit. A pin attribute called pin_is_clock_to_data is set to true for these pins.

You can set the ideal latency and ideal transition time of a clock by using the *set_clock_latency* and *set_clock_transition* commands.

See Also

- create_clock
- create_generated_clock
- set_clock_latency
- set_clock_transition

---

enable_enhanced_physical_multibit_banking

Data Types

Boolean

Default false

Description

When set to true before the *identify_register_banks* command, will switch to a new optimization engine that produces equal or better bits-per-cell with better timing QoR. The new engine will also implement the netlist changes immediately during the execution of *identify_register_banks*, thus creating an empty output script with a comment communicating that. The behavior of the new engine can be fully controlled by the use of the *set_multibit_options* command.

Feedback

See Also

- compile_enable_physical_multibit_banking

---

enable_golden_upf

Enables the golden UPF mode when set to true.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

true in DC Explorer

Group

mv

Description

This variable, when set to true, enables the golden UPF mode. By default, the variable is set to false and the golden UPF mode is disabled. In that case, the *save_upf* command writes out a full set of UPF commands that reflect the changes made by the tool to the UPF power intent of the design, as well as the UPF commands run during the tool session.

In the golden UPF mode, the same original "golden" UPF script file is used throughout the synthesis, physical implementation, and verification flow. The *save_upf* command writes out a supplemental UPF file that reflects only the changes made by the tool to the UPF power intent of the design. Downstream tools and verification tools use the golden and supplemental UPF files together to completely specify the power intent of the design. To use the golden UPF mode, set this variable to true before you execute any UPF commands. Once enabled, the golden UPF mode requires that you run UPF commands only by using the *load_upf* command. You cannot execute individual UPF commands at the shell prompt, except for UPF query commands.

In the golden UPF mode, the *save_upf* command writes out a supplemental UPF file, which contains only the UPF changes made by the tool during the session, such as tool-derived power intent and addition of power management cells. You can choose to write out a concise or verbose supplemental UPF file. A verbose file contains *connect_supply_net* commands that connect supply nets to the power management cells; these commands are needed if the netlist written in the session does not include PG connections. If the netlist includes PG connections, you can write out a concise supplemental UPF file instead.

Feedback

To determine the current value of this variable, use *printvar enable_golden_upf* command. For a list of all mv variables and their current values, use the *print_variable_group mv* command.

See Also

- load_upf
- save_upf
- upf_name_map

---

enable_instances_in_report_net

Enables the *report_net* command to report on instances in the current design.

Data Types

Boolean

Default true

Group

none

Description

This variable enables the *report_net* command to report on instances in the current design, when set to *true* (the default value).

When this variable is set to *false*, the *report_net* command reports only nets for the current design. Also, nets in the current instance are reported if the *current_instance* command is set.

See Also

- report_net

---

enable_keep_signal

Determines whether the tool forces preservation of a signal throughout compilation.

Data Types

Boolean

Default false

Feedback

Description

When you set the *enable_keep_signal* variable to *true*, the tool preserves marked nets throughout the compile process.

To mark a net to be preserved, the net should have the keep_signal_name label on it in the RTL source, and the *hdlin_keep_signal_name* variable should be set to either *user* or *user_driving*, depending on the nets to be preserved. See the *hdlin_keep_signal_name* variable man page for more information.

At the beginning of the compile process, the tool issues an OPT-154 message, indicating the presence of these marked nets. Note that preserving a net can cause QoR degradation.

Qor Impact:

- Logic connected to the preserved net is marked size_only.
- Automatic ungrouping of user hierarchies is disabled if the hierarchy has a preserved net connected to one of its ports.
- Constant propagation is disabled through preserved nets because the nets are marked dont_touch.
- Equal and opposite net optimization is disabled for preserved nets.
- The deletion of unused logic is disabled on the input cone of a preserved net.

Limitations:

- Ungrouping of synthetic cells during compile can cause the loss of a preserved net connected to its outputs (for example, two output nets of a synthetic cell that end up being driven by the same internal driver after ungrouping of the synthetic cell).
- Requests to preserve nets inside mapped synthetic cells post compile are ignored. Use a dont_touch explicitly on the synthetic cell hierarchy.
- Requests to preserve nets inside hierarchical clock-gating cells are ignored. Use a dont_touch explicitly on the clock-gating cell.

See Also

- hdlin_keep_signal_name
- enable_keep_signal_dt_net
- set_dont_touch

Feedback

enable_keep_signal_dt_net

Determines whether the tool forces preservation of a signal throughout compilation.

Data Types

Boolean

Default false

Description

When you set the *enable_keep_signal_dt_net* variable to *true*, the tool preserves marked nets during compilation.

Set the *enable_keep_signal_dt_net* variable to *true* before you use the *set_dont_touch* command on the net to mark it for preservation. Setting the *dont_touch* attribute on a net sets an implicit size_only on logic connected to that net even if logic connected to it is unmapped and not combinational.

At the beginning of compilation, the tool issues an OPT-154 warning indicating the presence of nets mark for preservation throughout compile. Preserving specific nets throughout compilation can cause quality of results (QoR) degradation. If you preserve a net that is in the critical path, the QoR degradation can be severe. This net preservation functionality is intended to facilitate verification. It should not be used to achieve final QoR goals.

Qor Impact:

- Logic connected to the preserved net is marked size_only.
- Automatic ungrouping of user hierarchies is disabled if the hierarchy has a preserved net connected to one of its ports.
- Constant propagation is disabled through preserved nets because the nets are marked dont_touch.
- Equal and opposite net optimization is disabled for preserved nets.
- The deletion of unused logic is disabled on the input cone of a preserved net.

Feedback

Limitations:

- Ungrouping of synthetic cells during compile can cause the loss of a preserved net connected to its outputs (for example, two output nets of a synthetic cell that end up being driven by the same internal driver after ungrouping of the synthetic cell).
- Requests to preserve nets inside mapped synthetic cells post compile are ignored. Use a dont_touch explicitly on the synthetic cell hierarchy.
- Requests to preserve nets inside hierarchical clock-gating cells are ignored. Use a dont_touch explicitly on the clock-gating cell.

See Also

- enable_keep_signal
- set_dont_touch

---

enable_nldm_timing_noise_signoff

Determines whether to use timing data and nonlinear delay filters to check for timing data and nonlinear delay errors in your library models.

Data Types

Boolean

Default 0

Description

The *enable_nldm_timing_noise_signoff* variable determines whether to use the timing data and the nonlinear delay filters to check for timing data and nonlinear delay errors in the library model.

enable_nldm_timing_noise_signoff is default false, Library Compiler wouldn't enable the filters for timing data and nonlinear delay errors.

If it is set to true, the Library Compiler command, read_lib <library_name> -signoff_screening, would enable the filters. Screener functions would be enabled separately by setting:

```
set chk_noise_exis true:
  Checks for missing noise data.
set chk_noise_range true:
  Checks noise table with a less than recommended range for
  input noise width and height.
set chk_noise_extpol true:
  Checks extrapolation on noise data.
set chk_noise_polar true:
```

```
  Checks for polarity on I-V curves.
set chk_IV true:
  Checks for polarity on I-V curves.
set chk_DC true:
  Screens negative noise tables.
set chk_NIC true:
  Screens noise immunity curves.
set chk_NPT true:
  Screens noise propagation tables.
set chk_reg1 true:
  Screens low and high noise regions.
set chk_reg2 true:
  Screens below_low and above_high noise regions.
set chk_noise_mono true:
  Screens non-monotonic noise arcs.
```

enable_page_mode

Controls whether long reports are displayed one page at a time (similar to the UNIX *more* command).

Data Types

Boolean

Default true

Description

This variable, when set to *true*, displays long reports one page at a time (similar to the UNIX *more* command). Commands affected by this variable include *list*, *help*, and the *report* commands.

To determine the current value of this variable, use the *list enable_page_mode* command. For a list of all system variables and their current values, use the *print_variable_group system* command.

enable_phys_lib_during_elab

Enables linking of physical libraries when you use the *analyze* and *elaborate* commands

Data Types

Boolean

Default false

Group

none

Feedback

Description

When this variable is set to *true*, the tool manually links physical libraries before using the *analyze* and *elaborate* commands. Then, the *analyze* and *elaborate* flow behaves the same as the *read_verilog* flow.

See Also

- analyze
- elaborate
- read_verilog

---

enable_presynthesis_floorplanning

Enables floorplan exploration on a design containing unmapped logic.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

true in DC Explorer

Description

This variable allows floorplan exploration on a design that contains unmapped logic. When it is set to true, DC Explorer creates dummy physical library cells for any cell in the design that does not have one, including unmapped cells, unmapped DesignWare components, black box cells, and so on.

See Also

- start_icc_dp

---

enable_recovery_removal_arcs

Controls whether the tool accepts recovery and removal arcs specified in the technology library.

Data Types

Boolean

Default false

Feedback

Description

This variable, when set to *true*, enables the acceptance of recovery and removal arcs specified in the technology library. Recovery or removal timing arcs impose constraints on asynchronous pins of sequential cells. Typically, recovery time specifies the time the inactive edge of the asynchronous signal has to arrive before the closing edge of the clock. Removal time specifies the length of time the active phase of the asynchronous signal must be held after the closing edge of clock.

To enable the *compile*, *report_timing*, and *report_constraint* commands to accept recovery or removal arcs specified in the library, set *enable_recovery_removal_arcs* to *true*.

Note that independent of the value of this variable, the *write_timing* and *report_delay_calculation* commands always accept and report recovery or removal timing information.

This variable is the logical opposite of the variable *timing_disable_recovery_removal_checks*. If you set either one of these variables to *true*, the tool automatically sets the other variable to *false*, and vice versa.

See Also

- compile
- timing_disable_recovery_removal_checks

---

enable_rule_based_query

Enables or disables rule-based matching.

Data Types

Boolean

Default false

Group

none

Description

When this variable is set to *true*, rule-based matching is enabled. However, you must run the *set_query_rules* command first. If you set this variable to *true* without running the *set_query_rules* command first, the default query rules are used.

When this variable is set to *true*, the runtime for the query might be slower. Disable rule-based matching when it is no longer needed.

Feedback

See Also

- set_query_rules

---

enable_slew_degradation

Determines whether the transition degradation is considered for nets with physical information.

Data Types

Boolean

Default true

Description

When this variable is set to *true* (the default value), timing calculation takes into account transition degradation across a net. For a long net, this might mean that the transition at the net driver could be very different than the various load pins. Because all load pins on the same net might not have the same transition time, optimization with physical information fixes transitions on a per pin basis. This might lead to increased runtime for these commands.

Setting this variable to *false* causes the timing calculation to not calculate transition degradation across a net. However, some libraries do have a transition degradation table that shows how to degrade the transition across a net. If slew degradation is not enabled, then the library transition degradation tables are used unless the *disable_library_transition_degradation* variable is set to *true*.

If neither slew degradation nor library transition degradation is enabled, the transition at the net driver will be the same transition at the load pin. Because all load pins on the same net have the same transition time, optimization with physical information attempts to fix transitions on a per net basis.

Regardless of the setting of this variable, slew degradation does not occur for multidriven nets. For multidriven nets, there can be cases of too much pessimism.

Regardless of the setting of this variable, slew degradation can only occur if there is either delay back-annotation for the net or physical locations for the pins of the net. Otherwise, the transition time calculation at the driver pin is too inaccurate to be degraded.

To determine the current value of this variable, use the *printvar enable_slew_degradation* command. For a list of all timing variables and their current values, use the *print_variable_group timing* command.

Feedback

See Also

- disable_library_transition_degradation

---

enable_special_level_shifter_naming

Enables special naming for automatically-inserted level shifters.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Group

mv

Description

When this variable is set to *true*, automatically-inserted level shifters by the *insert_level_shifters*, *compile*, and other commands are specially named. The name uses the following template:

<*prefix*> + <*PD OR Design name*> + "_LS" + #

The <*prefix*> is a user-specified prefix specified by the *level_shifter_naming_prefix* variable. The # is a number internally generated to keep this name unique. The <*PD OR Design Name*> is the name of the power domain where the level shifter is being added, or the design name if the power domain is not defined.

See Also

- level_shifter_naming_prefix

---

estimate_io_latency

Uses estimated I/O latency in timing calculations for ports when set to *true*.

Data Types

Boolean

Default false

Feedback

Group

timing

Description

This variable uses estimated I/O latency in timing calculations for ports when the value is set to *true*.

For each input port the I/O latency for max case is defined as the minimum clock network latency for the clocks in the fanout set of the port. The I/O latency for min case is the maximum clock latency in the fanout set.

For each output port the I/O latency for max case is defined as the maximum clock network latency for the clocks in the fanin set of the port. The I/O latency for min case is the minimum clock latency in the fanin set.

The I/O latency is not added to external delay in timing calculations if the *network_delay_included* switch is used. The ideal clocks are not used for I/O latency calculation.

This variable is best used after clock tree synthesis.

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

See Also

- report_timing
- set_clock_latency
- set_propagated_clock

---

exit_delete_command_log_file

Controls whether the file specified by the *command_log_file* variable is deleted after *design_analyzer* or dc_shell exits normally.

Data Types

string

Default false

Description

When this variable is set to *true*, the file specified by the *command_log_file* variable is deleted after *design_analyzer* or dc_shell exits normally. The default value is *false*. Set *exit_delete_command_log_file* to *false* to retain the file.

Feedback

To determine the current value of this variable, use the *printvar exit_delete_command_log_file* command. For a list of all system variables and their current values, use *print_variable_group system*.

See Also

- command_log_file

---

exit_delete_filename_log_file

Controls whether the file specified by the *filename_log_file* variable is deleted after *design_analyzer* or dc_shell exits normally.

Data Types

string

Default true

Description

When this variable is set to *true* (the default value), the file specified by the *filename_log_file* variable is deleted after *design_analyzer* or dc_shell exits normally. Set *exit_delete_filename_log_file* to *false* to retain the file.

To determine the current value of this variable, use the *printvar exit_delete_filename_log_file* command. For a list of all system variables and their current values, use *print_variable_group system*.

See Also

- filename_log_file

---

extract_max_parallel_computations

Sets the maximum degree of parallelism in multi-core extraction.

Data Types

Integer

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

0 in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Feedback

Group

physopt

Description

This application variable specifies the degree of parallelism in multi-core extraction. Increasing parallelism results in reduced runtime.

When set to 0, the number of cores used for extraction is specified directly by the value of the set_host_options command with the -max_cores option value.

When set to 1, multicore extraction is disabled.

When the value is smaller than the -max_cores value of the set_host_options command, the number of cores being used will not exceed this limit.

```
prompt> set extract_max_parallel_computations 4
prompt> extract_rc

EKL_MT: max_num_of_threads = 4
EKL_MT: total threadable CPU 1.74(= 0.44 + 0.44 + 0.43 + 0.43) seconds
EKL_MT: elapsed time 0 seconds
```

See Also

- set_host_options
- extract_rc

---

f

---

fanin_fanout_trace_arcs

Specifies the type of combinational arcs to trace during *all_fanin* and *all_fanout*.

Data Types

string

Description

This variable specifies the type of combinational arcs to trace during *all_fanin* and *all_fanout* commands, when that command's *-trace_arcs* option is not used.

Allowed values are *timing*, which permits tracing only of valid timing arcs (that is, arcs which are neither disabled nor invalid due to case analysis); and *all*, which permits tracing of all combinational arcs regardless of either case analysis or arc disabling.

Feedback

The default in DC/ICC is timing; the default in DC Explorer is all.

If the all_fanin/all_fanout command specifies option -trace_arcs, then that option takes effect regardless of the value of this variable.

See Also

- all_fanin
- all_fanout

filename_log_file

Specifies the name of the filename log file to be used in case a fatal error occurs during execution of *design_analyzer* or dc_shell.

Data Types

string

Default filenames.log

Description

This variable specifies the name of the filename log file to be used in case a fatal error occurs during execution of *design_analyzer* or dc_shell. The file specified by *filename_log_file* contains all filenames read in by *design_analyzer* or dc_shell, including .db, script, Verilog, VHDL, and include files for one invocation of the program. If there is a fatal error, you can easily identify the data files needed to reproduce the fatal error. If this variable is not specified, the default filename *filenames.log* is used. This file is deleted if the program exits normally, unless *exit_delete_filename_log_file* is set to *false*.

If the application has been invoked using the *-no_log* switch, then the process ID of the application and the timestamp is appended to the filename log file in the following format:

filenames_<*process_id*>_<*timestamp*>.log.

To determine the current value of this variable, use the *printvar filename_log_file* command. For a list of all system variables and their current values, use *print_variable_group system*.

See Also

- exit_delete_filename_log_file

find_allow_only_non_hier_ports

Instructs the *find* command to search for ports in subdesigns.

Feedback

Data Types

Boolean

Default false

Group

none

Description

When this variable is set to *true*, the *find* command searches for top-level ports only, ignoring ports in subdesigns.

By default, the *find port* command will also fetch ports in the subdesigns.

For example, if cell *c* is an instance of design *SUB1*, and this variable is set to *false* the command *find port c/A* fetches the port *A* on design *SUB1*. With the variable set to *true*, the command *find port c/A* results in a warning message as shown below:

```
prompt> printvar find_allow_only_non_hier_ports
find_allow_only_non_hier_ports = "false"

prompt> find port c/A
{"c/A"}

prompt> find_allow_only_non_hier_ports = true
"true"

prompt> printvar find_allow_only_non_hier_ports
find_allow_only_non_hier_ports = "true"

prompt> find port c/A
Warning: Can't find port 'c/A' in design 'SUB1'. (UID-95)
```

This variable is used to facilitate the netlist editing commands *connect_net* and *disconnect_net* which allow net and pin instances in addition to handling nets and pins in the current design.

See Also

- connect_net
- disconnect_net
- find
- get_ports
- remove_port

Feedback

find_converts_name_lists

Controls whether the *find* command converts the *name_list* string to a list of strings before searching for design objects.

Data Types

Boolean

Default false

Group

system_variables

Description

When this variable is set to *true*, the *find* command converts the *name_list* string to a list of strings before searching for design objects. In addition, when this variable is *true*, all commands that use the implicit *find* will convert appropriate strings to lists of strings before searching for objects. For example, *current_instance* uses the implicit *find* command and would convert the string *instance* to a list of strings before searching for objects.

The *find_converts_name_lists* variable provides backward compatibility with the premodification *find* command. When the variable is *false* (the default value), *find* executes according to its postmodification behavior; that is, strings are not converted to lists of strings.

To determine the current value of this variable, use the *printvar find_converts_name_lists* command. For a list of all system variables and their current values, use *print_variable_group system*.

See Also

- current_instance
- find

find_ignore_case

Controls whether the *find* command is case-sensitive when matching object names.

Data Types

Boolean

Default false

Feedback

Description

Normally, the *find* command is case-sensitive when matching names of objects. That is, for either an explicit or implicit invocation of the *find* command, *find* only matches objects whose names have the same case as the specified string.

When this variable is set to *true*, the *find* command performs case-insensitive string comparisons. This variable affects both implicit and explicit invocations of the *find* command.

To determine the current value of this variable, use the *list find_ignore_case* command. If the variable has not yet been defined, the *list* command will fail, indicating that the variable is undefined.

Use this variable only when necessary, as it may slightly increase the runtime of *find* operations.

See Also

- find

---

focalopt_power_critical_range

Specifies the slack threshold for performing final stage leakage-power recovery with the *focal_opt -power* command.

Data Types

float

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

0 in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

This variable specifies the power critical range in design timing units. The power critical range sets the slack threshold for performing final stage leakage-power recovery. You can specify a positive number, a negative number, or zero.

If the worst slack of all the paths through a cell is greater than the specified value, the *focal_opt -power* command performs final stage leakage-power recovery for that cell. Final stage leakage-power recovery preserves the timing QoR such that the resulting slack is greater than or equal to the specified value.

If a path has a slack value less than the specified value, none of the cells in that path are optimized.

By default, the power critical range value is zero, and leakage-power recovery is performed only those cells with positive slack (cells on nonviolating paths).

When you set this variable to a positive number, the command preserves timing QoR with positive slack and performs less leakage-power recovery. When you set this variable to a negative number, the command performs more aggressive leakage-power recovery and allows timing degradation.

This variable applies only to the *focal_opt -power* command. This variable applies to all scenarios in the design.

Examples

The following example defines the power critical range to be 10. If the timing unit for the design is picoseconds, this value means 10ps. The *focal_opt -power* command performs leakage-power recovery on cells with a worst slack greater than 10ps and maintains a worst slack of at least 10ps.

```
prompt> set_app_var focalopt_power_critical_range 10
```

The following example defines the power critical range to be -10. If the timing unit for the design is picoseconds, this value means -10ps. The *focal_opt -power* command performs leakage-power recovery on cells with a worst slack greater than -10ps and can degrade the worst slack up to -10ps.

```
prompt> set_app_var focalopt_power_critical_range -10
```

---

fsm_auto_inferring

Determines whether or not to automatically extract the finite state machine during compile.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable determines whether the compiler performs automatic extraction of finite state machines (FSM) during *compile_ultra*. If extraction of state machine was performed on a design previously, the compiler does not perform extraction again with the same encoding style.

Feedback

Examples

In the Presto flow, the compiler detects the FSM attributes from the HDL code and extracts the FSM during *compile_ultra*, as shown in the following example:

```
prompt> set_app_var fsm_auto_inferring true
prompt> read -f verilog example.v
prompt> set_fsm_encoding_style one_hot
prompt> compile_ultra
```

See Also

- set_fsm_encoding
- set_fsm_state_vector
- fsm_enable_state_minimization
- fsm_export_formality_state_info

---

fsm_enable_state_minimization

Determines whether or not the state minimization is performed for all finite state machines (FSMs) in the design.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

When this variable is set to *true*, the compiler performs the state minimization after the state extraction.

See Also

- fsm_auto_inferring
- fsm_export_formality_state_info

Feedback

fsm_export_formality_state_info

Determines whether or not state machine encoding information is exported into the files that will be used by Formality.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

When this variable is set to *true*, the state encoding information before and after finite state machine extraction is exported into two files, <*module_name*>.ref and <*module_name*>.imp, which will be used for Formality verification. The default value for this variable is *false*.

See Also

- fsm_auto_inferring
- fsm_enable_state_minimization

g

gen_bussing_exact_implicit

Controls whether schematics generated using the *create_schematic -implicit* command should contain implicit bus names instead of bus rippers.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Feedback

Description

When this variable is set to *true*, it specifies that schematics generated using the *create_schematic -implicit* command should contain no bus rippers. Instead, all bused connections are to be shown with implicit bus names.

Use this variable with the *-implicit* command-line option. By default, schematics generated using the *-implicit* option have rippers between any bus connections where the ripper connects to a pin in a column adjacent to the originating bus pin. Bused connections between cell pins more than one column away from each other are always shown disconnected with the *-implicit* option.

To determine the current value of this variable, use the *printvar gen_bussing_exact_implicit* command. For a list of all schematic variables and their current values, use *print_variable_group schematic*.

---

gen_cell_pin_name_separator

Specifies the character used to separate cell names and pin names in the bus names generated by the *create_schematic* command.

Data Types

string

Default / in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

If this variable is set, then its value is used to separate the cell and pin names in the bus names generated by *create_schematic*. By default, a / (slash) is used to separate the cell and pin names, thus creating bus names such as U0/OUT[0:3].

To determine the current value of this variable, use the *printvar gen_cell_pin_name_separator* command. For a list of all schematic variables and their current values, use *print_variable_group schematic*.

---

gen_create_netlist_busses

Controls whether *create_schematic* creates netlist buses whenever it creates buses on the schematic.

Data Types

Boolean

Default true in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

Controls whether the *create_schematic* command creates netlist buses whenever it creates buses on the schematic. When the value is *true* (the default), the *create_schematic* command creates netlist buses. Usually this occurs when *create_schematic* creates buses to connect to bused pins on the schematic. But it might also occur when there are bused ports on the schematic.

To determine the current value of this variable, use the *printvar gen_create_netlist_busses* command. For a list of all *schematic* variables and their current values, use *print_variable_group schematic*.

---

gen_dont_show_single_bit_busses

Controls whether single-bit buses are generated in the schematic.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable is used in conjunction with the *gen_show_created_busses* variable. When *gen_show_created_busses* is set to *true* and *gen_dont_show_single_bit_busses* is also set to *true*, single-bit buses created by gen are not printed out. This suppresses messages about the creation of single-bit buses in the schematic.

To determine the current value of this variable, use the *printvar gen_dont_show_single_bit_busses* command. For a list of all schematic variables and their current values, use the *print_variable_group schematic* command.

Feedback

See Also

- create_bus

---

gen_match_ripper_wire_widths

Controls whether the *create_schematic* command generates rippers whose width always equals the width of the ripped net.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable controls whether the *create_schematic* command generates rippers whose width always equals the width of the ripped net. When set to *true*, any rippers whose wire ends connect to scalar nets are of unit width.

By default, the *create_schematic* command connects scalar nets to the wire ends of multibit rippers. In this case, all of the concerned bits of the ripper are assumed to be shorted together and connected to that scalar net.

To determine the current value of this variable, use the *printvar gen_match_ripper_wire_widths* command. For a list of all *schematic* variables and their current values, use *print_variable_group schematic*.

---

gen_max_compound_name_length

Controls the maximum length of compound names of bus bundles for the *create_schematic -sge* command.

Data Types

integer

Default 256 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable controls the maximum length of compound names of bus bundles for the *create_schematic -sge* command. The default length is 256, which is the maximum length supported by SGE. Any buses with names longer than this variable are decomposed into their individual members in the schematic. If any such buses connect to cells referencing library symbols, those library symbols are ignored and new gen-created symbols are used. Any buses that these gen-created symbols have whose compound names exceed this length are blown up into their individual members.

To determine the current value of this variable, use the *printvar gen_max_compound_name_length* command. For a list of all schematic variables and their current values, use the *print_variable_group schematic* command.

---

gen_max_ports_on_symbol_side

Specifies the maximum allowed size of a symbol created by the *create_schematic* command.

Data Types

integer

Default 0 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable specifies the maximum allowed size of a symbol created by the *create_schematic* command. For example, if this variable is set to 5, symbols with no more than 5 ports on any 1 side are created. If this variable is not set or is set to 0, all input ports are placed on the left side of the symbol and all inout and output ports are placed on the right side.

To determine the current value of this variable, use the *printvar gen_max_ports_on_symbol_side* command. For a list of all *schematic* variables and their current values, use the *print_variable_group schematic* command.

---

gen_open_name_postfix

Specifies the postfix to be used by *create_schematic -sge* when creating placeholder net names for unconnected pins.

Feedback

Data Types

Boolean

Default "" in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

Specifies the postfix to be used by *create_schematic -sge* when creating placeholder net names for unconnected pins. The default is "".

The format of the net names is "%s%d%s", where the first "%s" is replaced by the value of *gen_open_name_prefix*, the second "%s" is replaced by the value of *gen_open_name_postfix*, and the "%d" is replaced by an integer whose value is generated automatically by *create_schematic -sge*.

For example, if *gen_open_name_prefix* = "Open", and *gen_open_name_postfix* = "_net", then the names created by *create_schematic* would be "Open1_net", "Open2_net", and so on.

The default values for *gen_open_name_prefix* and for *gen_open_name_postfix* are "Open" and "", respectively, so the default names created by *create_schematic* are Open1", "Open2", and so on.

To determine the current value of this variable, type *printvar gen_open_name_postfix*. For a list of all *schematic* variables and their current values, type *print_variable_group schematic*.

See Also

- gen_open_name_prefix

---

gen_open_name_prefix

Specifies the prefix to be used by *create_schematic -sge* when creating placeholder net names for unconnected pins.

Data Types

string

Default Open in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Feedback

Variable is not supported in DC Explorer

Description

Specifies the prefix to be used by *create_schematic -sge* when creating placeholder net names for unconnected pins. The default is "Open".

The format of the net names is "%s%d%s", where the first "%s" is replaced by the value of *gen_open_name_prefix*, the second "%s" is replaced by the value of *gen_open_name_postfix*, and the "%d" is replaced by an integer whose value is generated automatically by *create_schematic -sge*.

For example, if *gen_open_name_prefix* = "Open", and *gen_open_name_postfix* = "_net", then the names created by *create_schematic* would be "Open1_net", "Open2_net", and so on.

The default values for *gen_open_name_prefix* and for *gen_open_name_postfix* are "Open" and "", respectively, so the default names created by *create_schematic* are Open1", "Open2", and so on.

To determine the current value of this variable, type *printvar gen_open_name_prefix*. For a list of all *schematic* variables and their current values, type *print_variable_group schematic*.

See Also

- gen_open_name_postfix

---

gen_show_created_busses

Controls whether a message is printed out every time a schematic bus is created from cell pins for which no equivalent net bus exists in the netlist.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Group

schematic_variables

Feedback

Description

When true, a message is printed out every time a schematic bus is created from cell pins for which no equivalent net bus exists in the netlist. The default is false.

To determine the current value of this variable, type *printvar gen_show_created_busses*. For a list of all *schematic* variables and their current values, type *print_variable_group schematic*.

---

gen_show_created_symbols

Controls whether *create_schematic* prints a warning message every time it generates a new symbol for a cell because an appropriate symbol could not be found in the symbol libraries.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

Controls whether *create_schematic* prints a warning message every time it generates a new symbol for a cell because an appropriate symbol could not be found in the symbol libraries. The default is false.

To determine the current value of this variable, use *printvar gen_show_created_symbols*. For a list of all *schematic* variables and their current values, use the *print_variable_group schematic* command.

---

gen_single_osc_per_name

Controls whether more than one off-sheet connector with any particular name is drawn on any schematic sheet.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Feedback

Description

When this variable is set to true, only one off-sheet connector with any particular name is drawn on any schematic sheet. In cases where there could potentially be more than one off-sheet connector with the same name on any schematic page, only one connector is drawn and the others just have their net segments show up as unconnected stubs.

To determine the current value of this variable use *printvar gen_single_osc_per_name*. For a list of all *schematic* variables and their current values, use the *print_variable_group schematic* command.

---

generic_symbol_library

Specifies the generic symbol library used for schematics.

Data Types

string

Default generic.sdb in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable specifies the .db file that contains generic symbols, templates, and layers used for schematics.

To determine the current value of this variable, use the *printvar generic_symbol_library* command. For a list of all schematic variables and their current values, use the *print_variable_group schematic* command.

---

golden_upf_report_missing_objects

Enables reporting of missing objects during Golden UPF reapplication.

Data Types

Boolean

Default false

Group

mv

Feedback

Description

Setting this variable to true enables reporting of all missing objects when you reapply the golden UPF file to the design with the *load_upf* command.

In the golden UPF flow, the synthesis and physical implementation tools can change or remove objects in the design due to optimization, grouping, ungrouping, or expansion of wildcard names. Therefore, when you reapply the same golden UPF to the modified design, missing objects are expected as a normal part in the flow.

For this reason, by default, the tool does not report missing objects during reapplication of the golden UPF to the design, except for missing control signals identified by the *set_isolation_control*, *set_retention_control* and *create_power_switch* commands. These signals are not expected to be changed or removed by synthesis or physical implementation tools.

To enable reporting of all missing objects under these conditions, set the *golden_upf_report_missing_objects* to true. In that case, during reapplication of the golden UPF with the *load_upf* command, all missing objects are reported.

To determine the current value of this variable, use *printvar golden_upf_report_missing_objects* command. For a list of all mv variables and their current values, use the *print_variable_group mv* command.

See Also

- load_upf
- save_upf
- enable_golden_upf
- upf_name_map

---

gui_analyze_rtl_logic_level_threshold

Sets the logic-level threshold value used to determine bin coloring in a logic-level histogram.

Data Types

integer

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

0 in DC Explorer

Feedback

Group

None

Description

The *gui_analyze_rtl_logic_level_threshold* variable allows you to set a single threshold value between 1 and infinity for the timing paths in a logic-level histogram. If this variable is not set or is set to 0, the tool uses the technology library to determine individual threshold values for each path.

See Also

- gui_analyze_rtl_paths

---

gui_analyze_rtl_paths

Controls the number of timing paths that the tool uses to determine which bins should be colored red in a logic-level histogram.

Data Types

integer

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

10 in DC Explorer

Group

None

Description

The *gui_analyze_rtl_paths* variable specifies the minimum number of timing paths that the tool uses to color bins red in a logic-level histogram. The default value is 10.

Starting with the bin that contains the paths with the most logic levels, the tool evaluates the paths in each bin from right to left. If the number of logic levels on at least one path exceeds its threshold value, the bin is colored red. The tool continues to color the bins red until it reaches the *gui_analyze_rtl_paths* number of paths. After reaching this value, the tool colors the bins white until it finds a bin in which none of the paths contain more logic levels than their threshold values. The rest of the bins are colored green.

See Also

- gui_analyze_rtl_logic_level_threshold

Feedback

gui_online_browser

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

Data Types

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
```

h

hdl_keep_licenses

Controls whether HDL licenses that are checked out remain checked out throughout the tool's session or are released after use.

Data Types

Boolean

Default true

Description

When this variable is set to *true* (the default value), HDL licenses that are checked out remain checked out throughout the tool's session. When this variable is *false*, HDL licenses are released during the execution of the *compile* command, after *compile* completes the subtasks that require the license.

To determine the current value of this variable, use the *printvar hdl_keep_licenses* command. For a list of all HDL variables and their current values, use *print_variable_group hdl*.

---

hdl_preferred_license

Selects an HDL license to check out, if none is currently checked out.

Data Types

string

Default ""

Description

This variable selects an HDL license to check out, if none is currently checked out. Allowed values are *vhdl*, which selects the VHDL Compiler license or *verilog*, which selects the HDL Compiler license. The elaboration process requires an HDL license.

To determine the current value of this variable, use the *printvar hdl_preferred_license* command. For a list of all HDL variables and their current values, use *print_variable_group hdl*.

---

hdlin_allow_unpacked_array_concat_on_port

Controls whether concat with unpacked array operand(s) in instance port expression is allowed. Effective only for Presto HDL Compiler.

Data Types

Boolean

Default false

Group

hdl_variables

Description

Controls whether concat with unpacked array operand(s) in instance port expression is allowed, such as *b[3:2]* in *.A({a, b[3:2], c})* shown in the following sample test case:

```
module top (
    input  logic a,
    input  logic b[3:2],
    input  logic c,
    output logic d
);
```

Feedback

```
sub sub(.A({a, b[3:2], c}), .B(d));

endmodule

module sub(
    input  logic A[3:0],
    output logic B
);
  ...
endmodule
```

where, *b[3:2]* in the instance port expression *{a, b[3:2], c}* is an unpacked array.

The default setting is *false*, meaning that Presto does not allow the above case, and an error (*ELAB-929*) will be issued for the above case.

When *hdlin_allow_unpacked_array_concat_on_port* is turned on, Presto will allow it with a warning message (*ELAB-1991*) issued.

As a precaution, you should examine the RTL code to make sure the connection is as expected. For example, in the corresponding port declaration in module *sub*, the port *A* should also be declared as an unpacked array, instead of a packed array such as an integer or bit-vector.

See Also

- ELAB-929
- ELAB-1991

---

hdlin_always_fsm_complete

Set attribute:FSM_COMPLETE for all of FSMs

Data Types

Boolean

Default true

Description

When this option is set true, all of FSMs will have attribute: FSM_COMPLETE. Its default value is false.

See Also

- hdlin_failsafe_fsm

Feedback

hdlin_analyze_prioritize_command_line_defines

Enables prioritization of command-line define macros, disabling their redefinition.

Data Types

Boolean

Default true

Description

The *hdlin_analyze_prioritize_command_line_defines* application option allows the user to prioritize command-line defines, disabling their redefinition from RTL code. If disabled, higher-precedence defines, for example, the ones defined in the command line or -vcs files can be redefined. For example:

```
`define VALB 4
  module test (output [3:0] b);
    assign b=`VALB;
  endmodule

  // Pass defines from analyze command
  analyze -define "VALB=7" -format sverilog test.sv
  elaborate test

  write -format verilog -hier -output net.v

  Netlist:
  assign b[3] = 1'b0;
  assign b[2] = 1'b1;
  assign b[1] = 1'b1;
  assign b[0] = 1'b1;
```

See Also

- analyze

hdlin_analyze_verbose_mode

Queries information about preprocessing of Verilog and SystemVerilog conditional compilation compiler directives and macro expansions.

Data Types

integer

Default 0

Feedback

Description

The *hdlin_analyze_verbose_mode* variable lets you query information about preprocessing of the Verilog or SystemVerilog RTL, including macro expansions and conditional compilation compiler directives. You use this information to debug design issues, especially for designs with a large number of macros. To query the preprocessing information about conditional compilation compiler directives such as `ifdef, set this variable to 1. To query about these and additionally macro expansions, set the variable to 2. The default is 0.

The informational messages can be found in the log by looking for the message code (VER-7). For example,

```
Information: ./example.v:6: Analyzing `ifdef then clause because MYRTL is
 defined. (VER-7)
Information: ./example.v:7: Macro |`MYMACRO| expanded to |1'b0|. (VER-7)
Information: ./example.v:8: Skipping `else clause. (VER-7)
```

See Also

- analyze

---

hdlin_auto_save_templates

Controls whether HDL designs containing parameters are read in as templates.

Data Types

Boolean

Default false

Description

When set to *true* this variable reads in HDL designs containing parameters as templates. HDL parameter files include Verilog modules with parameter declarations and VHDL entities with generic declarations.

When an HDL file is read in as a template, it can be manipulated with the *printvar -templates*, *remove_template*, and *elaborate* commands. The template can also be built automatically when instantiated from another HDL design.

When the value is *false* (the default), HDL designs are read in and built instantly. This can be overridden for a design with a template pseudo comment.

For Verilog, place the following comment anywhere in the module:

```
// synopsys template
```

For VHDL, place the following comment in the entity after the *port* declaration:

```
-- synopsys template
```

To determine the current value of this variable, use the *printvar hdlin_auto_save_templates* command. For a list of HDL variables and their current values, use the *print_variable_group hdl* command.

See Also

- elaborate
- template_parameter_style
- template_separator_style

---

hdlin_check_no_latch

Controls whether a warning message is issued for latches inferred by incomplete combinational assignments.

Data Types

Boolean

Default false

Description

This variable enables a warning message for latches inferred by incomplete combinational assignments.

The following SystemVerilog RTL shows two examples of incomplete combinational assignment:

```
always @(*) begin
  if (en) begin
   out1 = in1;
  end  // no assignment for (!en)
end

always @(*)
  case(in2)
    2'h0: out2= 10'b001;
    2'h1: out2= 10'b010;
    2'h2: out2= 10'b100;
    // no assignment for (in2 == 2'h3)
  endcase
```

Feedback

In both cases, the assignment is incomplete and the tool must implement latches to hold the current value when the unassigned condition occurs.

The tool always prints sequential inference messages for these latch cells:

```
Inferred memory devices in process
        in routine top line 5 in file
                '/proj/rtl/top.v'.
=============================================================================
===
|    Register Name    | Type  | Width | Bus | MB | AR | AS | SR | SS | ST
 |
=============================================================================
===
|       q1_reg        | Latch |   1   |  N  | N  | N  | N  | -  | -  | -
 |
=============================================================================
===
```

But when the *hdlin_check_no_latch* variable is set to *true*, the tool also issues an ELAB-395 warning message:

```
Inferred memory devices in process
        in routine top line 5 in file
                '/proj/rtl/top.v'.
=============================================================================
===
|    Register Name    | Type  | Width | Bus | MB | AR | AS | SR | SS | ST
 |
=============================================================================
===
|       q1_reg        | Latch |   1   |  N  | N  | N  | N  | -  | -  | -
 |
=============================================================================
===
Warning:  Latch inferred in design top read with
 'hdlin_check_no_latch' (ELAB-395)
```

If your design should not have any incomplete combinational assignments, add the following commands to your script prior to RTL read:

```
set_app_var hdlin_check_no_latch true  ;# the default is false
set_message_info -id ELAB-395 -stop_on  ;# force an error when this
 message occurs
```

If incomplete combinational assignments are found, the RTL read will fail:

```
Inferred memory devices in process
        in routine top line 5 in file
                '/proj/rtl/top.v'.
=============================================================================
===
```

```
|    Register Name    | Type   | Width | Bus | MB | AR | AS | SR | SS | ST
 |
=========================================================================
===
|       q1_reg        | Latch |   1   |  N  | N  | N  | N  | -  | -  | -
 |
=========================================================================
===
Severe Error:  Latch inferred in design top read with
 'hdlin_check_no_latch' (ELAB-395)
Presto compilation completed successfully.
Current design is now '/proj/rtl/top.db:top'
Loaded 1 design.
Current design is 'top'.
Error: A Severe error has occurred. To ensure that the script does not
 continue,
the value of sh_continue_on_error has been overridden to be false. Your
 script
is being interrupted. To see the Tcl call stack for the part of your
 script which
generated the Severe error use the error_info command.   (CMD-103)
Error: Severe error encountered
        Use error_info for more info. (CMD-013)
```

The default for this variable is *false*, which disables the ELAB-395 message (but does not prevent the inferred latch cells).

To determine the current value of this variable, use the *printvar hdlin_check_no_latch* command. For a list of HDL variables and their current values, use *print_variable_group hdl*.

See Also

- ELAB-395

---

hdlin_elab_errors_deep

Specifies debug mode to clean RTL with elaboration, link, and internal errors.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

true in DC Explorer

Feedback

Group

hdl_variables

Description

This variable allows you to be in debug mode to clean RTL with elaboration errors, link errors, and internal errors, similar to DC Explorer.

HDL Compiler elaborates a design in a top-down hierarchical order. By default, this variable is *false* (unless using DC Explorer), so the elaboration failure of a certain module prohibits the elaboration of all child modules. This behavior is safe but less informative.

When this variable is set to *true* (as in DC Explorer), HDL Compiler continues to elaborate its child modules even if a module's elaboration fails. In this way, HDL Compiler can elaborate as many modules as possible in a single elaboration run. Thus, more RTL code elaboration problems can be detected in a single run, which improves efficiency.

This variable works for both the *elaborate* and *read* command.

To benefit from this variable, the designs must be free from syntax errors during the analyze process.

In Design Compiler, in *hdlin_elab_errors_deep* mode, no designs are created after elaboration. So commands such as *current_design* or *link* do not work in this debug mode.

In DC Explorer, even in *hdlin_elab_errors_deep* mode (default), designs are created after elaboration. So commands such as *current_design* or *link* works in this debug mode.

---

hdlin_elaborate_black_box

Specify one or more modules names criterias to have the corresponding module body ignored during design elaboration.

Data Types

string

Default ""

Description

The *hdlin_elaborate_black_box* application variable allows the user to specify one or more modules names criterias, including wildcard matching, to have the corresponding body ignored during elaboration.

The modules names criterias are specified by setting the application variable *hdlin_elaborate_black_box* to a list of modules, e.g.

```
prompt> set_app_var hdlin_elaborate_black_box "mod1 mod2 ..."
```

Feedback

Here mod1, mod2 are names of modules as coded in the RTL.

The warning message (ELAB-748) is generated during the *elaborate* or *read* command. An example of the message is shown below:

```
Warning:  ./test.sv:521: The body of module 'mod1' is being discarded,
 because the module name meets hdlin_elaborate_black_box criteria.
 (ELAB-748)
```

The valid module names are those coded in the RTL and the design names, e.g. those reported by *list_designs*, giving priority to the design names.

If a modules RTL name meets hdlin_elaborate_black criteria and its design name meets hdlin_elaborate_black_box_all_except criteria, its body will not be ignored.

If a modules design name meets hdlin_elaborate_black_box criteria and its RTL name meets hdlin_elaborate_black_box_all_except criteria, its body will be ignored.

No messages are generated for specified names that do not meet valid module names.

See Also

- elaborate
- read
- list_designs
- hdlin_elaborate_black_box_all_except

---

hdlin_elaborate_black_box_all_except

Specify one or more modules names criterias to not ignore the body of during design elaboration.

Data Types

string

Default ""

Description

The *hdlin_elaborate_black_box_all_except* application variable allows the user to specify one or more modules names criterias, including wildcard matching, to not ignore the corresponding body of during elaboration.

The modules names criterias are specified by setting the application variable *hdlin_elaborate_black_box_all_except* to a list of modules, e.g.

```
prompt> set_app_var hdlin_elaborate_black_box_all_except "top mod1  ..."
```

Feedback

Here top, mod1, mod2 are names of modules as coded in the RTL.

The warning message (ELAB-748) is generated during the *elaborate* or *read* command. An example of the message is shown below:

```
Warning:  ./test.sv:521: The body of module 'mod2' is being discarded,
 because the module name is not in hdlin_elaborate_black_box_all_except.
 (ELAB-748)
```

The valid module names are those coded in the RTL and the design names, e.g. those reported by *list_designs*, giving priority to the design names.

If a modules RTL name meets hdlin_elaborate_black criteria and its design name meets hdlin_elaborate_black_box_all_except criteria, its body will not be ignored.

If a modules design name meets hdlin_elaborate_black_box criteria and its RTL name meets hdlin_elaborate_black_box_all_except criteria, its body will be ignored.

No messages are generated for specified names that do not match valid module names.

See Also

- elaborate
- read
- list_designs
- hdlin_elaborate_black_box

---

hdlin_enable_assertions

Controls the tool's use of SystemVerilog Assertions.

Data Types

Boolean

Default false

Group

hdl_variables

Description

Setting *hdlin_enable_assertions* to *true* allows the *elaborate* and *read_sverilog* commands to process a synthesis-friendly subset of SystemVerilog RTL Assertions. Facts added via assertions can contribute to optimization by ruling out machine states that cannot arise during normal operation of the design. This switch controls actions taken only during the elaborate phase, including elaborations that occur during linkage or compilation.

To be activated, an assertion statement must belong to a narrow category of synthesizable deferred immediate assertions. It must have a dedicated statement label that gives it a unique, lexically scoped name. It also must be analyzed in *sverilog* format (or by *read_sverilog*).

The asserted expression should compare only one variable or signal to a compile-time constant, or else it should make a $onehot or $onehot0 claim about a collection of signal wires.

Consult the *HDL Compiler for SystemVerilog User Guide* for the exact definition of the subset currently supported by this release. All labeled assertions within this subset survive analysis,

To confirm that an assertion should be considered during optimization do the following:

- Set *hdlin_enable_assertions true* when it is being elaborated.
- Set the lexically scoped name of its statement label as an element of the *confirmed_SVA()* array variable in the tool environment in which the assertion is elaborated. Set it to the value of *true*.

When the above conditions are met, the labeled and confirmed assertions encoded in the intermediate (or template) design files can begin to influence logic minimization done by the tool.

Note that either linking or elaborating designs with unresolved template references can cause subdesign elaborations whose confirmed assertions may also be exploited by the tool's optimizations. The following is an example that confirms 3 assertions named *assertion_label1*, *global_assertion1*, and *assumption_about_port*:

```
analyze -f sverilog my_file.sv
# The following lines can be brought in via a source command
set confirmed_SVA(module1.scope1.assertion_label1) true
set confirmed_SVA(module1.global_assertion1) true
set confirmed_SVA(module2.generated_scope[0].assumption_about_port) true

set hdlin_enable_assertions true
elaborate module2
```

See Also

- elaborate
- ELAB-333
- ELAB-414

Feedback

hdlin_enable_configurations

Controls the configuration support by the tool.

Data Types

Boolean

Default false

Group

hdl_variables

Description

The *hdlin_enable_configurations* variable enables configuration support by the tool when set to *true*. The default value is *false*.

To specify a configuration, use the *analyze* command. To elaborate an entity with a specific configuration, use the *elaborate* command with the configuration name as the design name. For example, if file.vhdl contains the configuration named *my_configuration*, read and elaborate the design as follows:

```
prompt> set hdlin_enable_configurations true
prompt> analyze -f vhdl file.vhdl
prompt> elaborate my_configuration
```

The *read* command ignores configurations.

Setting this variable to *true* entails more stringent link-time checks. If designs do not use configurations, it is advised not to turn on this switch.

For more information refer to the *HDL Compiler for VHDL User Guide*.

See Also

- analyze
- elaborate

hdlin_enable_elaborate_ref_linking

Controls the hierarchical elaboration flow support by SystemVerilog.

Data Types

Boolean

Default false

Feedback

Group

hdl_variables

Description

This variable, when set to *true*, enables hierarchical elaboration support by HDL Compiler SystemVerilog. It allows the *elaborate* command to include additional contextual linkage information embedded into the elaborate designs, such as typeparams, parameter, and interface specializations.

Later, during final top module elaboration, the switch allows the linker to reach and analyze this additional contextual linkage information stored in designs, helping the pre-elaborated designs to be matched with the top module instantiations, enabling the hierarchical elaboration in a bottom-up flow.

For more information, see the *HDL Compiler for SystemVerilog User Guide*.

See Also

- elaborate

---

hdlin_enable_elaborate_update

Reanalyzes out-of-date intermediate files if the source can be found.

Data Types

Boolean

Default true

Description

This variable should be set to *true* before the *analyze* command is run.

---

hdlin_enable_hier_map

Enable HDL Compiler to generate *guide_hier_map* guidance in SVF.

Data Types

Boolean

Default false

Description

This flow is available for cases where differences in logical hierarchy inference between Design Compiler and Formality causes mis-application of SVF constraints leading to a failure in verification.

For these cases, Design Compiler can issue a different style of SVF guidance with respect to the design hierarchy than the usual flow. Design Compiler produces a set of *guide_hier_map* guidances to capture a baseline for the logical hierarchy, before switching to the regular *guide_instance_map* guidances for any further changes to the logical hierarchy.

This variable must be set to *true* prior to any design read commands (*analyze*, *elaborate*, *read_verilog*, *read_file*, etc.) This variable setting will enable the *set_verification_top* command (see below for usage) and suppress *guide_instance_map* guidances that are normally generated in (V)HDL Compiler.

This flow also requires that the *set_verification_top* command be added to your design read flow. This command must be run after the final *elaborate* command of your design read step, and prior to any further steps such as *compile_ultra*, grouping or ungrouping, etc.

This command generates the *guide_hier_map* guidances as the first guidances in the SVF file. Then subsequent SVF guidances will be consistent with this logical hierarchy baseline.

If *hdlin_enable_hier_map* is set to *true* and no *set_verification_top* command is issued before SVF generation is terminated, an error message is issued.

Examples

The following script shows the correct command ordering for this flow.

```
# Specify SVF output file
prompt> set_svf ./guidance/top.svf

# Enable GHM flow
prompt> set_app_var hdlin_enable_hier_map true

# Read design files
prompt> analyze -format sverilog {sub.sv top.sv}
prompt> elaborate top
prompt> current_design top

# Issue this command after reading last RTL file and before
# other commands that can modify netlist
prompt> set_verification_top
```

Feedback

See Also

- analyze
- elaborate
- read_verilog
- read_file
- set_svf
- set_verification_top

---

hdlin_enable_hier_naming

Controls if the registers and instances in a design are named with their hierarchical path attached or not.

Data Types

Boolean

Default false

Description

This variable controls if registers and instances are named in the hierarchical way or not. For example,

```
4 generate for (i=0; i < 2; i = i + 1 ) begin : mygenblk
      5   always @(posedge clk or negedge rst) begin : myblk
      6     reg myreg;
      7     if( ! rst ) begin
      8       qout = 0;
      9       myreg = 0;
     10     end
     11     else begin
     12       qout = myreg;
     13       myreg = datain;
     14     end
     15   end
     16 end endgenerate
```

If *hdlin_enable_hier_naming* is set to *true*, the registers inferred from the code on line 6 will be 'mygenblk[0].myblk.myreg_reg' and 'mygenblk[1].myblk.myreg_reg'.

To determine the current value of this variable, use the *printvar hdlin_enable_hier_naming* command. For a list of HDL variables and their current values, use *print_variable_group hdl*.

Feedback

hdlin_enable_ieee_1735_support

Data Types

Boolean

Default true

Description

Setting this variable to *true* enables support in the analyze command for the decryption of Verilog and SystemVerilog encrypted envelopes in the style of IEEE Std 1735-2014. IEEE std 1735-2014 for VHDL is enabled by default and cannot be controlled by this setting.

hdlin_enable_persistent_macros

Controls whether a macro definition encountered during a Verilog or SystemVerilog analyze command is remembered in subsequent Verilog or SystemVerilog analyze commands.

Data Types

Boolean

Default false

Group

hdl_variables

Description

By default, each Verilog and SystemVerilog analyze command is independent, except for the intermediate files that are saved into the libraries defined by define_design_lib.

When the hdlin_enable_persistent_macros application variable is enabled, the analyze command as its first action compiles the file of macro definitions specified by the hdlin_persistent_macros_filename application variable (if the file exists) and then as its last action it overwrites that file with the current macro definitions (or creates the file if it doesn't exist yet).

For example, if the analyze command encounters `undef directives or macro redefinitions, those changes will be reflected in the file at the end of the analyze command.

Like the intermediate files that are saved into libraries, the macro definitions file is protected with Synopsys encryption.

It is recommended to remove any macros definition file that might be left over from a previous session by doing the following before the first analyze command.

```
file delete -force [get_app_var hdlin_persistent_macros_filename]
```

See Also

- hdlin_persistent_macros_filename

---

hdlin_enable_persistent_sv_interfaces

Controls whether a System Verilog Interface declaration encountered during a SystemVerilog analyze command is remembered in subsequent SystemVerilog analyze commands.

Data Types

Boolean

Default false

Group

hdl_variables

Description

By default, each Verilog and SystemVerilog analyze command is independent, except for the intermediate files that are saved into the libraries defined by define_design_lib.

When the hdlin_enable_persistent_sv_interfaces application variable is enabled, the analyze command as its first action compiles the file of macro definitions specified by the hdlin_persistent_sv_interfaces_filename application variable (if the file exists) and then as its last action it overwrites that file with the current macro definitions (or creates the file if it doesn't exist yet).

Like the intermediate files that are saved into libraries, the macro definitions file is protected with Synopsys encryption.

It is recommended to remove any interface declaration file that might be left over from a previous session by doing the following before the first analyze command.

```
file delete -force [get_app_var hdlin_persistent_sv_interfaces_filename]
```

See Also

- hdlin_persistent_sv_interfaces_filename

---

hdlin_enable_relative_placement

Enables RTL relative placement support.

Feedback

Data Types

string

Default rb

Description

This variable determines whether and how the tool supports RTL relative placement. Valid values for *hdlin_enable_relative_placement* are *mux*, *rb*, *rb;mux*, *mux;rb*, and *none*.

- When the value is *mux*, relative placement is enabled for MUX_OPs.
- When the value is *rb*, relative placement is enabled for register banks.
- When the value is *rb;mux* or *mux;rb*, relative placement is enabled for register banks and MUX_OPs.
- When the value is *none*, the tool disables relative placement for all of the structures.

Relative placement for MUX_OPs is not supported if it violates the *hdlin_mux_rp_limit* variable.

To determine the current value of this variable, use the *printvar hdlin_enable_relative_placement* command. For a list of HDL variables and their current values, use *print_variable_group hdl*.

See Also

- hdlin_mux_rp_limit

---

hdlin_enable_rtldrc_info

Controls whether RTL TestDRC creates file name and line number information for HDL constructs and instances for designs processed by subsequent shell commands.

Data Types

Boolean

Default false

Group

hdl_variables

Description

This variable controls whether RTL TestDRC creates file name and line number information for HDL constructs and instances for designs processed by subsequent shell commands.

In the Synopsys RTL Test Design Rule Checking (TestDRC) tool, this information must be created for the designs in order to view the links between GTECH and mapped designs and HDL code. This analysis information is created when you process designs using the *dft_drc* shell command.

When set to *false* (the default), no file or line information is created.

---

hdlin_enable_upf_compatible_naming

Controls HDLC naming style settings to make it easier to apply the same UPF file across multiple tools at the RTL level.

Data Types

Boolean

Default false

Description

This variable controls HDLC naming style settings to make it easier to apply the same UPF file across multiple tools at the RTL level. This will typically make it conform more closely to language standard naming conventions.

When *hdlin_enable_upf_compatible_naming* is set to *true*, it will have the following effects:

```
set hdlin_enable_hier_naming true
set hdlin_field_naming_style "%s.%s"
set hdlin_vhdl_preserve_case true
```

When *hdlin_enable_upf_compatible_naming* is set to *false* (the default), it will respect the current settings of the above variables.

To determine the current value of this variable, use the *printvar hdlin_enable_upf_compatible_naming* command. For a list of HDL variables and their current values, use *print_variable_group hdl*.

See Also

- hdlin_enable_hier_naming
- hdlin_field_naming_style
- hdlin_vhdl_preserve_case

---

hdlin_failsafe_fsm

Enable Failsafe Finite State Machine support

Feedback

Data Types

Boolean

Default false

Description

When this option is set true, Failsafe Finite State Machine will be enabled. Its default value is false.

See Also

- hdlin_always_fsm_complete

---

hdlin_ff_always_async_set_reset

Controls whether the tool checks and reports asynchronous set and reset conditions of flip-flops.

Data Types

Boolean

Default true

Group

hdl_variables

Description

This variable, when set to *true* instructs the tool to check and report asynchronous set and reset conditions of flip-flops. Setting this variable to *false* disables this behavior.

Set this variable to *false* if you do not use asynchronous set or reset conditions, if you do not want asynchronous set or reset devices inferred, or if you want your design to be input faster.

The *async_set_reset* pragma can be used to overwrite this variable. See the *HDL Compiler for Verilog User Guide* or the *HDL Compiler for VHDL User Guide* for more details about the pragma.

To determine the current value of this variable, use the *printvar hdlin_ff_always_async_set_reset* command. For a list of HDL variables and their current values, use *print_variable_group hdl*.

Feedback

hdlin_ff_always_sync_set_reset

Controls whether every constant 0 loaded on a flip-flop under the clock event is used for synchronous reset, and every constant 1 loaded on a flip-flop under the clock event is used for synchronous set.

Data Types

Boolean

Default false

Description

This variable controls whether every constant 0 loaded on a flip-flop under the clock event is used for synchronous reset, and every constant 1 loaded on a flip-flop under the clock event is used for synchronous set. Setting the value to *true* for a design subsequently analyzed, results in this behavior.

For more details, see the the *HDL Compiler for Verilog User Guide* or the *HDL Compiler for VHDL User Guide*.

To determine the current value of this variable, use the *printvar hdlin_ff_always_sync_set_reset* command. For a list of HDL variables and their current values, use *print_variable_group hdl*.

hdlin_field_naming_style

Defines the parts of the net names that the tool generates corresponding to the fields in VHDL records or in SystemVerilog structs.

Data Types

string

Default ""

Description

This variable defines the parts of the net names that the tool generates corresponding to the fields in VHDL records and SystemVerilog structs. By default, the *hdlin_field_naming_style* is derived from the *bus_naming_style* and *bus_dimension_separator_style*. But when these are "%s[%d]" and "][" or are "%s<%d>" and "><", then it is possible to specify an independent *hdlin_field_naming_style*, such as "%s<%s>", "%s[%s]", or "%s.%s", in which the first %s stands for the name up to the field and the second %s stands for the field. The *hdlin_field_naming_style* must be of the form "%sX%s" or "%sX%sY", where X and Y are (possibly identical) non-whitespace characters.

Feedback

See Also

- bus_naming_style

---

hdlin_generate_naming_style

Specifies the naming style for *generated* design instances in VHDL designs.

Data Types

string

Default %s_%d

Description

This variable specifies the naming style for *generated* design instances. For example,

```
gen: for i in 0 to 1 generate
   U: CELL port map(a(i), z(i));
end generate gen;
```

If *hdlin_generate_naming_style* is set to *%s_%d* (the default), the example results in instances named U_0 and U_1.

To determine the current value of this variable, use the *printvar hdlin_generate_naming_style* command. For a list of HDL variables and their current values, use *print_variable_group hdl*.

---

hdlin_generate_operator_sharing_data

Specifies if presto should generate the operator sharing data.

Data Types

Boolean

Default false

Description

This variable specifies if the presto should generate data about operators that can be shared. By default presto does not generate this data to save runtie. This variable can be set to true to generate the operator sharing data. In DC Expert mode, this data is used by compile to share operators if possible.

---

hdlin_generate_separator_style

Specifies the separator string for instances generated in multiple-nested loops.

Feedback

Data Types

string

Default _

Description

This variable specifies the separator string for instances generated in multiple-nested loops. For example;

```
L1: for I in 3 downto 0 generate
   L2: for J in I downto 0 generate
      U: MY_AND port map ( A, B, Z);
   end generate;
end generate;
```

If *hdlin_generate_separator_style* is set to __ (the default is _), the example results in instances named U_3__3, U_3__2, U_3__1, U_3__0, U_2__2, U_2__1, U_2__0, U_1__1, U_1__0, and U_0__0.

To determine the current value of this variable, use the *printvar hdlin_generate_separator_style* command. For a list of HDL variables and their current values, use *print_variable_group hdl*.

The value of this variable affects only designs read in VHDL format.

---

hdlin_ignore_ghm_errors

Controls whether the tool ignores errors in the *set_verification_top* verification guidance flow.

Data Types

Boolean

Default false

Description

This variable controls whether the tool requires that the *set_verification_top* command be run after reading in RTL files.

The *set_verification_top* command is part of the recommended verification guidance creation flow, which is described in the *hdlin_enable_hier_map* man page.

By default, the tool requires that the *set_verification_top* command be run after reading all RTL files but prior to any design modification commands. This ensures that correct verification guidance is created.

If needed, you can disable this requirement by setting the *hdlin_ignore_ghm_errors* application variable to *false*. However, be aware that this might result in incorrect guidance.

This variable has no effect when the *hdlin_enable_hier_map* variable is set to its default of *false*, which results in the legacy verification guidance creation flow.

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

See Also

- set_verification_top
- hdlin_enable_hier_map

---

hdlin_infer_complex_set_reset

Controls whether the two commands *analyze* and *elaborate* attempt to infer complex set/reset logic for sequential cells.

Data Types

Boolean

Default false

Description

Controls whether the two commands *analyze* and *elaborate* attempt to infer complex set/reset logic for sequential cells.

By default Presto HDL Compiler will not attempt to infer set/reset logic that it estimates to be 'complex', for example, driven by some other instance or with many gates between the input and the sequential cell. And the complex set/reset logic will stay on the data line of the sequential cell.

For quality of result it's usually desirable for the set/reset logic to be fast and simple, but when this is not so, setting the variable to true tells Presto HDL Compiler to consider some additional, more complex possibilities, although potentially at the cost of increased runtime and memory usage.

The variable works for both Verilog and VHDL input files.

See Also

- analyze
- elaborate

Feedback

hdlin_infer_function_local_latches

Controls whether the HDL Compiler infers latches inside functions and tasks.

Data Types

Boolean

Default false

Group

hdl_variables

Description

This variable controls whether the HDL Compiler infers latches inside functions and tasks.

When the value of this variable is *true*, latches inside function and task bodies can be inferred, according to the same rules used in always blocks.

When the value is *false* (the default), no latches are inferred in functions or tasks.

hdlin_infer_local_sync_enable_only

Controls which controls are fed into synch_enable pins and which are fed into ne xt_state pins.

Data Types

Boolean

Default false

Group

hdl_variables

Description

When the value of this variable is *true* and a particular coding style has been used, HDL Compiler produces a netlist in which the control signals local to sequential blocks feed into synch_enable pins, while external ones feed into next_state pins.

hdlin_infer_multibit

Specifies inference of multibit components for an entire design.

Feedback

Data Types

string

Default default_none

Description

This variable specifies inference of multibit components for an entire design. The allowed values for the *hdlin_infer_multibit* variable are default_all, default_none, and never.

The never setting prevents inference of multibit components from HDL regardless of directives (Verilog) or attributes (VHDL).

The default_all setting infers multibit components on all bused registers except where directives or attributes indicate otherwise.

The default_none setting specifies that only attributes or directives are used to infer multibit components. This is the default for the *hdlin_infer_multibit* variable.

For details, see the *HDL Compiler for Verilog User Guide* or the *HDL Compiler for VHDL User Guide*.

To determine the current value of this variable, use the *printvar hdlin_infer_multibit* command. For a list of HDL variables and their current values, use *print_variable_group hdl*.

---

hdlin_infer_mux

Determines whether and how the tool infers a MUX_OP.

Data Types

string

Default default

Description

This variable determines whether and how the tool infers a MUX_OP. Allowed values for *hdlin_infer_mux* are *all*, *none*, and *default*.

When the value is *all*, the tool attempts to infer a MUX_OP for a signal or variable assigned in a case or if statement and for an expression using the ?: operator.

When the value is *none*, the tool does not attempt to infer any MUX_OPs for a Verilog or VHDL design.

When the value is *default*, the tool attempts to infer a MUX_OP for a case or if statement and for the ?: operator, when the statement is in a process associated with the *infer_mux*

attribute or directive. A MUX_OP cannot be inferred if it violates the *hdlin_mux_size_limit* variable.

Note that the *infer_mux_override* directive will force the MUX_OP inference, ignoring the *hdlin_infer_mux* setting.

For details, refer to the *HDL Compiler for Verilog User Guide* or the *HDL Compiler for VHDL User Guide*.

Note that *hdlin_infer_mux* only changes MUX_OP inference for if and case statements and ?: operations. It does not affect array read operations. For MUX_OP inference of array read operations see the *hdlin_mux_for_array_read_sparseness_limit* variable man page.

See Also

- hdlin_mux_for_array_read_sparseness_limit
- hdlin_mux_size_limit

---

hdlin_interface_port_ABI

Chooses a linkage convention for SystemVerilog ports of type interface or modport.

Data Types

integer

Default 2

Description

The *hdlin_interface_port_ABI* variable is effective only when HDL Compiler (Presto) parses the source code of an *interface* or *module* declaration in SystemVerilog format (*-f sverilog*) during an *analyze* or *read* command. Its setting selects the linkage conventions to be used when interface instances or ports connect across a subdesign port. Because this setting is recorded when the interface is analyzed, its author can determine interface-specific aspects the ABI conventions once for all users, independent of any other interfaces they might also use. Since the setting is also recorded by modules that form and pass along arrays of interface instances, it allows a module which creates arrays to specify the conventions for connecting them to its subdesigns.

The memory elements and signal wires synthesized specifically for each interface instance connect to physical input, output, or inout ports at every hierarchical boundary where that interface instance connects to other design elements through an interface port or modport. The "Application Binary Interface" (ABI) is a set of conventions for which connections to make, their positional order and port names, and how to treat the contents of arrayed interface ports. Both designs being connected must agree on the ABI of their actual and formal interface ports. This agreement occurs automatically when both interface clients

are elaborated using identical versions of the interface, analyzed with the same choice of ABI setting. Usually it suffices to analyze the interface once into a design library (such as "WORK") shared by all its clients as they are elaborated.

The integer value of *hdlin_interface_port_ABI* encodes a bit vector with one bit position for each ABI feature. Its default value is 3'b010.

bit[0] Canonical Bounds (ABI 3'b??0 vs 3'b??1)

Bit 0 only affects arrayed ports (of subdesign reference cells) whose elements are instances (or modports) of some interface definition. The ABI bit[0] of the current module controls, not the ABI of the interface or the down design. The effect is noticeable only when the interface array actual port expression has bounds that differ from the canonical bounds for unpacked arrays (such as [*n*] or [0:*n*-1]) and the actual port is passed by .*portname*(...). In these cases, the array index values are explicit in the port names being matched by Design Compiler's linker.

- An interface array connected from a module whose ABI value is *0* in bit position 0 (this is the default) uses the actual array expression's bounds to invent port names of the reference cell. Note that different instantiations of the same down design might then require distinct formal port names, and these will be produced by multiple elaborations of the down design.
- An interface array connected from a module whose ABI value is *1* in bit position 0 uses canonical bounds to invent port names of the reference cell. Note that part-selects of unpacked arrays always produce canonical bounds, so this option avoids the primary source of multiple elaboration due to bounds-matching.

  Note that with either setting, formal port names can differ from the internal net names to which they connect. In fully linked designs, this produces net names in the down design that are taken from their connectivity with the top design. The canonical setting exactly resembles the names produced during linkage for all other forms of bused ports

bit[1] Demoted Modports (ABI 3'b?0? vs 3'b?1?)

Bit 1 affects only those port connections where the formal interface port specifies a modport and the actual interface instance does not.

When the value in bit position 1 is *1* (the default), these connections are successfully linked per the IEEE-1800 standard definition. This is accomplished by adding dummy (unused) formal ports to a special elaboration of the design that correspond in name and position to all interface content that lies outside the given formal modport. This elaboration uses a different *-param* string than that produced for a design reference with matching modport specifications for this interface.

When the value in bit position 1 is *0*, no special elaboration is created to accommodate formal-only modport selection; no dummy formal ports are added. In general, this will

Feedback

prevent linkage of an actual interface instance to a formal modport selected by the down design.

Note that the default (demote) setting allows interface modport connections to be made using .* and .portname forms as the actual port. The cost of this convenience is the introduction of unused design ports.

bit[2] API-style (ABI 2'b0?? vs 3'b1??)

Bit positions 2 and above are reserved for future extensions of the interface port ABI. Premature use of these settings is not supported.

See Also

- analyze

---

hdlin_interface_port_downto

Controls the HDLC canonicalization of GTECH ports inferred from a SystemVerilog RTL arrayed interface port declaration.

Data Types

Boolean

Default false

Description

Ths variable controls the HDLC canonicalization of GTECH ports inferred from a SystemVerilog RTL arrayed interface port declaration.

When the variable is set to *false* (the default), the GTECH ports will be in a little-endian style with the leftmost ports indexed with 0. When the variable is set to *true*, the GTECH ports will instead be in big-endian style with the rightmost ports indexed with 0.

It is recommended that an entire SystemVerilog design either consistently declare all arrayed interface ports with the C-style [N] syntax (that means [0:N-1]) or consistently declare all of them as [N-1:0] but also enable this variable.

---

hdlin_intermediate_file_method

Specifies how HDL Compiler stores information about analyzed RTL files.

Data Types

string

Default legacy

Feedback

Description

This variable controls how the *analyze* command stores the data from analyzed RTL files, which is subsequently used by the *elaborate* command.

The *hdlin_intermediate_file_method* variable can be set as follows:

- *legacy* - The data is stored as multiple files (named after module, interfaces or packages names) in the directory defined by the *define_design_lib* command.
- *compact* - The data is stored in a single file (named "default.hdllib") in the directory defined by the *define_design_lib* command.
- *none* - The data is stored in memory instead of on disk.

For the values of *legacy* and *compact*, the data remains persistent and available for use by elaboration in subsequent sessions.

For the value of *none*, all RTL analysis and elaboration must be performed in a single session. However, runtime can improve significantly (versus storing files on disk) if the filesystem performance is poor.

If the *hdlin_auto_save_templates* application variable is set to *true*, then this variable also affects the *read_verilog* and *read_sverilog* commands.

This variable applies only to Verilog and SystemVerilog RTL analysis; it is not yet implemented for VHDL RTL analysis.

See Also

- analyze
- define_design_lib
- elaborate
- hdlin_auto_save_templates
- read_sverilog
- read_verilog
- read_file

---

hdlin_keep_signal_name

Determines whether HDL Compiler attempts to keep a signal name.

Data Types

string

Feedback

Default all_driving

Description

The *hdlin_keep_signal_name* variable is available only in HDL Compiler (Presto Verilog) and HDL Compiler (Presto VHDL). For this discussion, these tools will be referred to as Presto. The original HDL and VHDL compilers do not support *hdlin_keep_signal_name*.

The value of *hdlin_keep_signal_name* determines whether Presto attempts to keep a signal name. This includes preserving cells between the signal and the input or output port. The *hdlin_keep_signal_name* variable provides Presto with guideline information for keeping a signal name, but it is not intended to be a 100% guarantee for keeping a signal. In some Presto optimizations, such as removing redundant code, the signal name may be optimized away, even if the signal is associated with the *keep_signal_name* attribute or directive. The allowed values are *all*, *none*, *user*, *user_driving*, and *all_driving*, as described below. The *user* and *user_driving* values require the *keep_signal_name* directive.

none

Presto does not attempt to keep any signals. This value overrides the *keep_signal_name* directive.

all

Presto attempts to preserve a signal if the signal is not removed by optimizations. Dangling and driving nets are considered. This value does not guarantee that a signal is kept. The *all* value generally preserves fewer signals than the *user* value. For example, if there are 3 sequential nets named *N1*, *N2*, and *N3*, the *all* value may keep only *N1*. Setting *hdlin_keep_signal_name* to the value of *user* and setting the *keep_signal_name* directive on *N1*, *N2*, and *N3* generally keeps *N1*, *N2*, *N3*.

This value may cause the *check_design* command to issue LINT-2 and LINT-3 warnings:

```
Warning: In design '...', net '...' has no drivers.
Logic 0 assumed.  (LINT-3)

Warning: In design '...', net '...' driven by pin '
(no pin) ' has no loads.  (LINT-2)
```

all_driving

Presto attempts to preserve a signal if the signal is not removed by optimizations and the signal is in an output path. Only driving nets are considered. This value does not guarantee that a signal is kept. For signals that are not in an input path (do not have drivers), Presto connects the input to ground, assuming logic 0 (ground) for the driver.

user

This value works with the *keep_signal_name* directive. Presto attempts to preserve a signal if the signal is not removed by optimizations, and that signal is labeled with the *keep_signal_name* directive. Dangling and driving nets are considered. Although not guaranteed, Presto usually keeps the specified signal for this configuration.

user_driving

This value works with the *keep_signal_name* directive. Presto attempts to preserve a signal if the signal is not removed by optimizations, the signal is in an output path, and the signal is labeled with the *keep_signal_name* directive. Only driving nets are considered. Although not guaranteed, Presto usually keeps the specified signal for this configuration. For signals that are not in an input path (do not have drivers), Presto connects the input to ground, assuming logic 0 (ground) for the driver.

Examples

In Example 1, you use the *hdlin_keep_signal_name* variable and the *keep_signal_name* directive to control what signals Presto attempts to preserve.

```
Example 1 - Verilog

module test (a, b, T,  A, c);
input  [3:0]   a;
input  [7:0]   b;
input T;
input A;
output reg  [7:0] c;
wire d, e;
//synopsys async_set_reset "A"
//synopsys keep_signal_name "e"

assign e = ( a[3] & ~a[2] &  a[1] & ~a[0] );
assign d = ( a[3] & ~a[2] &  a[1] & ~a[0] );

always @(T or b or A or d)
if (A)
     c = 8'h0;
       else if (T & d)
     c = b;
endmodule
```

The internal wire e is not kept if the line //synopsys keep_signal_name "e" is removed, and *hdlin_keep_signal_name* is either *none*, *user*, *user_driving*, or *all_driving*. However, e is kept if *hdlin_keep_signal_name* is *all*. Also, e is kept if *hdlin_keep_signal_name* is *user* and //synopsys keep_signal_name is attached to "e".

In Example 2, by default Presto attempts to preserve the test1 and test2 signals because they are in output paths. Note that the test2 signal is not in an input path; Presto does not try to keep the test3 signal because it is not in an output path. Presto optimizes away the *syn1* and *syn2* signals.

```
Example 2 - Verilog

module test12 (in1, in2, in3, in4, out1,out2 );
  input  [3:0]   in1;
  input  [7:0]   in2;
  input in3;
  input in4;
  output reg  [7:0] out1,out2;

  wire test1,test2, test3, syn1,syn2;

//synopsys async_set_reset "in4"

  assign test1 = ( in1[3] & ~in1[2] & in1[1] & ~in1[0] );
 //test1 signal is in an input and output path
  assign test2 = syn1+syn2;
 //test2 signal is in a output path, but it is not in an input path
  assign test3 = in1 + in2;
 //test3 signal is in an input path, but it is not in an output path
  always @(in3 or in2 or in4 or test1)
  out2 = test2+out1;
  always @(in3 or in2 or in4 or test1)
      if (in4)
        out1 = 8'h0;
      else
      if (in3 & test1)
       out1 =  in2;
endmodule
```

One way to keep the test3 signal is to set *hdlin_keep_signal_name* to *user* and place the *keep_signal_name* directive on test3, as shown in Example 3.

```
Example 3 - Verilog

module test12 (in1, in2, in3, in4, out1,out2 );
  input  [3:0]   in1;
  input  [7:0]   in2;
  input in3;
  input in4;
  output reg  [7:0] out1,out2;

  wire test1,test2, test3, syn1,syn2;
//synopsys keep_signal_name "test3"

//synopsys async_set_reset "in4"

    assign test1 = ( in1[3] & ~in1[2] & in1[1] & ~in1[0] );
```

Feedback

```
   //test1 signal is in an input and output path
    assign test2 = syn1+syn2;
   //test2 signal is in a output path, but it is not in an input path
    assign test3 = in1 + in2;
   //test3 signal is in an input path, but it is not in an output path
    always @(in3 or in2 or in4 or test1)
    out2 = test2+out1;
    always @(in3 or in2 or in4 or test1)
        if (in4)
          out1 = 8'h0;
        else
        if (in3 & test1)
         out1 = in2;
endmodule
```

Table 1 shows how the variable settings affect the preservation of signal test3, with and without the directive applied to it. Asterisks (*) indicate that Presto does not attempt to keep the signal and might remove it.

Table 1 Variable and Directive Matrix for Signal test3

hdlin_keep_signal_name = all_driving	all	none	user	user_driving
keep_signal_name is not set on test3	attempts to keep	*	*	*
keep_signal_name is set on test3	attempts to keep	*	attempts to keep	*

Table 2 shows how the variable settings affect the preservation of signal test2, with and without the directive applied to it. Asterisks (*) indicate that Presto does not attempt to keep the signal and might remove it.

Table 2 Variable and Directive Matrix for Signal test2

hdlin_keep_signal_name = all_driving	all	none	user	user_driving
keep_signal_name is not set on test2	attempts to keep	*	*	* to keep
keep_signal_name is set on test2	attempts to keep	*	attempts to keep	attempts to keep

Feedback

Table 3 shows how the variable settings affect the preservation of signal test1, with and without the directive applied to it. Asterisks (*) indicate that Presto does not attempt to keep the signal and might remove it.

```
Table 3  Variable and Directive Matrix for Signal test1

hdlin_keep_signal_name =  all     none   user     user_driving
 all_driving
-------------------------------------------------------------------------
---
keep_signal_name is     attempts   *      *           *
 attempts
not set on test1        to keep                                 to keep

keep_signal_name is     attempts   *     attempts   attempts
 attempts
set on test1            to keep          to keep    to keep       to keep
-------------------------------------------------------------------------
---
```

To determine the current value of this variable, use the following command:

```
prompt> printvar hdlin_keep_signal_name
```

Use the following command to get a list of HDL variables and their current values in DB (dcsh) mode :

```
prompt> print_variable_group hdl
```

Use the following command to get a list of HDL variables and their current values in XG and DB (Tcl) mode:

```
prompt> print_variable_group
```

See Also

- enable_keep_signal

---

hdlin_latch_always_async_set_reset

Uses every constant 0 loaded on a latch for asynchronous reset and uses every constant 1 loaded on a latch for asynchronous set for a design subsequently analyzed.

Data Types

Boolean

Default false

Feedback

Description

This variable, when set to *true*, instructs the tool to use every constant 0 loaded on a latch for asynchronous reset, and to use every constant 1 loaded on a latch for asynchronous set, for a design subsequently analyzed.

The default value is *false*.

For more details, see the *HDL Compiler for Verilog User Guide* or the *HDL Compiler for VHDL User Guide*.

To determine the current value of this variable, use the *printvar hdlin_latch_always_async_set_reset* command. For a list of HDL variables and their current values, use *print_variable_group hdl*.

---

hdlin_legacy_naming

Data Types

Boolean

Default false

Description

In unified compile flow, names of registers, ports etc. conform to the language standard naming conventions.

When hdlin_legacy_naming is set to true, default value of hdlin_enable_upf_compatible_naming is set to false.

When hdlin_legacy_naming is set to false (the default), it will have the following effects:

```
set hdlin_enable_upf_compatible_naming true
set hdlin_enable_hier_naming true
set hdlin_field_naming_style "%s.%s"
```

To determine the current value of this variable, use the *printvar hdlin_legacy_naming* command. For a list of HDL variables and their current values, use *print_variable_group hdl*.

See Also

- hdlin_enable_hier_naming
- hdlin_field_naming_style

Feedback

hdlin_logfile_format

Configures the format of the log file.

Data Types

string

Default 202409

Group

hdl_variables

Description

Configures the log file to a particular format. Each format is represented by the tool version that introduced substantial changes to the reports and information messages. Valid values for *hdlin_logfile_format* are *202409* and *pre202409*.

hdlin_module_arch_name_splitting

Controls whether Presto HDL Compiler recognizes a special format of Verilog module names, which allows users to specify both a module and an implementation architecture.

Data Types

Boolean

Default false

Description

During an *analyze* or *read* command, when the *hdlin_module_arch_name_splitting* value is *true*, and a Verilog module name contains the special separator string "__" (two underscores), Presto HDL Compiler interprets the module name as specifying both a module and a specific architecture or implementation for that module. The portion of the original module name before the separator string becomes the new module name, and the portion after the string becomes the architecture.

For example, a module named *mod__impl* would be renamed to *mod* and marked as having architecture *impl*.

If *hdlin_module_arch_name_splitting* is set to *false* (the default), the renaming never occurs and modules always retain their original names. If the variable is set to *true*, the "__" is used to divide the module name into module and architecture as described above.

This capability exists to allow the creation of synthetic libraries using Verilog, with multiple architectures per module.

To determine the current value of this variable, use the *printvar hdlin_module_arch_name_splitting* command. For a list of HDL variables and their current values, use *print_variable_group hdl*.

---

hdlin_module_name_limit

Specifies the length threshold for compressing elaborated module names when the *hdlin_shorten_long_module_name* variable is set to *true*.

Data Types

integer

Default 2048

Description

This variable specifies the length threshold for whether the Presto HDL Compiler compresses names for elaborated modules when the *hdlin_shorten_long_module_name* variable is set to *true*.

When *hdlin_shorten_long_module_name* is *true*, if the name of an elaborated module is longer than the value of *hdlin_module_name_limit* (default 256), then a compressed name is used that is easier for downstream tools to handle. The initial part of the name is still recognizable, but the tail of the name is replaced with numbers.

To determine the current value of this variable, use the *printvar hdlin_module_name_limit* command. For a list of HDL variables and their current values, use *printvar_variable_group hdl*.

See Also

- hdlin_shorten_long_module_name

---

hdlin_module_param_id_length_threshold

Configures the threshold for showing an ID association for a set of non-default parameters during elaboration in log.

Data Types

string

Default 128

Feedback

Description

This option configures the threshold for the string that characterizes the non-default parameters of a module instantiation during elaboration. If the parameter string length is less than this option value, the *HDL-193* information message will show all parameters; on the other hand, if the string is greater, an ID will be created and associated with that set of parameters. The ID is numeric, incremental (starting from 1), and will be preserved for the same session. Consider that this feature is only enabled if *hdlin_logfile_format* option value is equal or greater than *202409*.

For example, consider the following parametrized module and a threshold of 5:

```
module sub #(parameter PARAM1 = 0, parameter PARAM2 = 0) ();
    initial begin
        $display("PARAM1: %d, PARAM2: %d", PARAM1, PARAM2);
    end
    endmodule

    module top;
        sub #(1, 2) u_sub1 ();
        sub #(.PARAM1(3), .PARAM2(4)) u_sub2 ();
    endmodule
```

If *top* module is elaborated, *u_sub1* will be instantiated with arguments "1,2", being a string length of 3; whereas *u_sub2* will be instantiated with the argument string "PARAM1=3,PARAM2=4", having a size (17) that exceeds the threshold, hence, an ID is associated.

---

hdlin_mux_for_array_read_sparseness_limit

Prevents inference of a sparse multiplexer for array read operations when the percentage of MUX_OP data inputs that are connected is below *hdlin_mux_for_array_read_sparseness_limit*.

Data Types

integer

Default 90

Description

This variable prevents inference of a sparse multiplexer for array read operations when the percentage of MUX_OP data inputs that are connected is below *hdlin_mux_for_array_read_sparseness_limit*.

For array read operations, HDL Compiler can infer a MUX_OP or a SELECT_OP. MUX_OPs can be mapped to MUX cells in the technology library but SEL_OPs cannot be mapped in this manner. MUX cells might be desirable for lowering congestion or

Feedback

implementing specific switching behavior. For an array read operation, if the array size is not a power of 2, HDL Compiler uses a MUX_OP whose size is the next power of 2. The unused data inputs are disconnected. This is called a sparse multiplexer.

The *hdlin_mux_for_array_read_sparseness_limit* variable guides the HDL Compiler to choose the right cell for an array read operation. This value corresponds to the lowest percentage of MUX_OP data inputs that must be connected. If the percentage is below this limit, HDL Compiler infers a SELECT_OP instead, whose number of data inputs does not need to be a power of 2, and therefore gives lower area.

The default percentage is 90. This means that if at least 90% of MUX_OP data inputs are connected, a MUX_OP is inferred. If this percentage is below 90%, a SELECT_OP is inferred instead.

The following are examples of code that will not infer a MUX_OP:

```
VHDL
----
signal a : bit_vector (0 to 99);
signal x: bit;
  ...
x <= a(i);

Verilog
-------
reg [0:99] a;
reg x;
  ...
x = a[i];
```

In both examples, a MUX_OP of 128 data inputs is required, but only 100 would be connected. This gives a percentage of 100/128*100; that is, only 78% of MUX_OP data inputs are connected. As this is below 90%, a SELECT_OP will be inferred.

Note that this variable prevents sparse MUX_OP inference for array read operations only. Inference of MUX_OP for if, case, and ?: operations is governed by the *hdlin_infer_mux*, *hdlin_mux_size_limit* and *hdlin_mux_oversize_ratio* variables.

See Also

- hdlin_infer_mux
- hdlin_mux_oversize_ratio
- hdlin_mux_size_limit

Feedback

hdlin_mux_oversize_ratio

Prevents inference of a sparse multiplexer, when the ratio of MUX_OP data inputs to unique data inputs is above the *hdlin_mux_oversize_ratio*.

Data Types

integer

Default 100

Description

This variable prevents inference of a sparse multiplexer, when the ratio of MUX_OP data inputs to unique data inputs is above the *hdlin_mux_oversize_ratio*.

The default ratio is 100.

Here are examples of code that infer a sparse MUX_OP:

```
VHDL
----
case sel is
  when "0001" =>   z <= data(3);
  when "0010" =>   z <= data(2);
  when "0100" =>   z <= data(1);
  when "1000" =>   z <= data(0);
  when others =>   z <= 'X';
end case;

Verilog
-------
case (sel)
  4'b0001: z = data[3];
  4'b0010: z = data[2];
  4'b0100: z = data[1];
  4'b1000: z = data[0];
  default: z = 1'bx;
endcase
```

When a MUX_OP is inferred from each of the previous examples, the mux oversize ratio is calculated. These examples infer a 16-to-1 MUX_OP, with 4 unique data inputs. The oversize ratio is 4. Setting *hdlin_mux_oversize_ratio* to 3 prevents either example from inferring a MUX_OP. A setting of 4 or greater allows these examples to infer a MUX_OP.

To determine the current value of this variable, use the *printvar hdlin_mux_oversize_ratio* command. For a list of HDL variables and their current values, use *print_variable_group hdl*.

Feedback

See Also

- hdlin_infer_mux
- hdlin_mux_size_limit

---

hdlin_mux_rp_limit

Limits the minimum bit width and minimum number of inputs for a relative placement multiplexer.

Data Types

string

Default 128x4

Description

This variable limits the minimum bit width and minimum number of inputs for a relative placement multiplexer.

The value is *mxn* where *m* is the minimum bit width and *n* is the minimum number of inputs for the multiplexer. The default value is 128x4, meaning that HDL Compiler does not generate relative placement MUX_OPs for multiplexers specified with fewer than 4 inputs and a bit width of less than 128.

The time it takes to process a relative placement MUX is proportional to the value of *hdlin_mux_rp_limit* squared. For example, the time it takes to process a 128x64 MUX for relative placement is approximately four times that for processing a 128x32 MUX.

To determine the current value of this variable, use the *printvar hdlin_mux_rp_limit* command. For a list of HDL variables and their current values, use *print_variable_group hdl*.

See Also

- hdlin_enable_relative_placement

---

hdlin_mux_size_limit

Limits the number of inputs of an inferred multiplexer.

Data Types

integer

Default 32

Feedback

Description

This variable limits the number of inputs of an inferred multiplexer. The default is value is 32. HDL Compiler does not generate MUX_OPs for multiplexers specified with more than the maximum number of 32 inputs. The size of a multiplexer can be impounded by nested if or case statements.

The time taken to process a MUX is proportional to the value of *hdlin_mux_size_limit* squared. Extreme care must be taken when setting the limit to a figure larger than 32. If the design appears to hang with a limit greater than 32, lower the limit and try again. There is an additional, nonpractical, internal limit of 1073741824 for the size of a inferred mux.

For details, refer to the *HDL Compiler for Verilog User Guide* or the *HDL Compiler for VHDL User Guide*.

To determine the current value of this variable, use the *printvar hdlin_mux_size_limit* command. For a list of HDL variables and their current values, use *print_variable_group hdl*.

See Also

- hdlin_infer_mux

---

hdlin_mux_size_min

Sets the lower bound for the number of inputs required to infer a multiplexer.

Data Types

integer

Default 2

Description

This variable sets the lower bound for the number of inputs required to infer a multiplexer. The default value is 2. HDL Compiler does not generate MUX_OPs for multiplexers specified with less than this minimum number of inputs.

To determine the current value of this variable, use the *printvar hdlin_mux_size_min* command. For a list of HDL variables and their current values, use *printvar_variable_group hdl*.

Feedback

See Also

- hdlin_infer_mux
- hdlin_mux_oversize_ratio
- hdlin_mux_size_limit

---

hdlin_mux_size_only

Controls which MUX_OP cells receive the *size_only* attribute in HDL Compiler.

Data Types

integer

Default 1

Description

To ensure that MUX_OP GTECH cells are mapped to MUX technology cells, you must apply a *size_only* attribute to the cells to prevent logic decomposition in later optimization steps. Beginning with the B-2008.09-SP3 release, you can control which MUX_OP cells receive the *size_only* attribute by using the *hdlin_mux_size_only* variable. The following options are valid for *hdlin_mux_size_only*:

0

Specifies that no cells receive the *size_only* attribute.

1

Specifies that all MUX_OP cells directly inferred by an RTL *infer_mux* pragma and that are on set and reset signals receive the *size_only* attribute.

2

Specifies that all MUX_OP cells directly inferred by an RTL *infer_mux* pragma receive the *size_only* attribute.

3

Specifies that all MUX_OP cells directly or indirectly inferred and that are on set and reset signals receive the *size_only* attribute.

4

Specifies that all MUX_OP cells directly or indirectly inferred receive the *size_only* attribute.

A MUX_OP is "directly inferred" when it results from an *infer_mux* pragma applied to a specific RTL statement or operator.

Feedback

A MUX_OP is "indirectly inferred" when it results from an *infer_mux* pragma on a named block or from the *hdlin_infer_mux* variable being set to *all*.

To determine the current value of this variable, use the *printvar hdlin_mux_size_only* command. For a list of HDL variables and their current values, use *printvar_variable_group hdl*.

Note that all MUX_OP cells inferred using the *infer_mux_override* directive will receive the *size_only* attribute.

See Also

- set_size_only
- hdlin_infer_mux

---

hdlin_naming_register_suffix_on_field

Enables suffix ("_reg"/"_tri"/"_tri_enable_reg") appending on the field of structured cells.

Data Types

Boolean

Default false

Description

The *hdlin_naming_register_suffix_on_field* application option allows the user to add the register suffix on the basename of the structure or as the last field.

Some examples of the change using this application option are:

```
false                                    true
out1_reg                                 out1_reg
out2_reg[1]                              out2[1]_reg
out3_tri[0][somefield]                   out3[0][somefield]_tri
out3_tri_enable_reg[0][somefield][1]
 out3[0][somefield]_tri_enable_reg[1]
```

See Also

- elaborate

---

hdlin_netlist_unloaded_signals

Controls whether Presto optimizes away the unloaded signals before netlisting.

Feedback

Data Types

Boolean

Default false

Group

hdl_variables

Description

This variable controls whether Presto optimizes away unloaded signals before netlisting.

When set to *true*, Presto netlists unloaded signals. The purpose is for Presto to report pruned registers accurately when these redundant netlist is removed later.

When set to *false* (the default), Presto optimizes away unloaded signals before netlisting. So Presto will not create the redundant logic first and then remove. The early optimization may cause missing report pruned registers.

See Also

- hdlin_report_sequential_pruning

---

hdlin_persistent_macros_filename

Controls the name of the file to remember macro definitions encountered during a Verilog or SystemVerilog analyze command in subsequent Verilog or SystemVerilog analyze commands when the hdlin_enable_persistent_macros application variable is enabled.

Data Types

string

Default syn_auto_generated_macro_file.sv

Group

hdl_variables

Description

By default, each Verilog and SystemVerilog analyze command is independent, except for the intermediate files that are saved into the libraries defined by define_design_lib.

When the hdlin_enable_persistent_macros application variable is enabled, the analyze command as its first action compiles the file of macro definitions specified by the hdlin_persistent_macros_filename application variable (if the file exists) and then as its

Feedback

last action it overwrites that file with the current macro definitions (or creates the file if it doesn't exist yet).

For example, if the analyze command encounters `undef directives or macro redefinitions, those changes will be reflected in the file at the end of the analyze command.

Like the intermediate files that are saved into libraries, the macro definitions file is protected with Synopsys encryption.

It is recommended to remove any macros definition file that might be left over from a previous session by doing the following before the first analyze command.

```
file delete -force [get_app_var hdlin_persistent_macros_filename]
```

See Also

- hdlin_enable_persistent_macros

---

hdlin_persistent_sv_interfaces_filename

Controls the name of the file to remember SystemVerilog interface definitions encountered during a SystemVerilog analyze command in subsequent SystemVerilog analyze commands when the hdlin_enable_persistent_sv_interfaces application variable is enabled.

Data Types

string

Default syn_auto_generated_interface_file.sv

Group

hdl_variables

Description

By default, each Verilog and SystemVerilog analyze command is independent, except for the intermediate files that are saved into the libraries defined by define_design_lib.

When the hdlin_enable_persistent_sv_interfaces application variable is enabled, the analyze command as its first action compiles the file of macro definitions specified by the hdlin_persistent_sv_interfaces_filename application variable (if the file exists) and then as its last action it overwrites that file with the current macro definitions (or creates the file if it doesn't exist yet).

Like the intermediate files that are saved into libraries, the macro definitions file is protected with Synopsys encryption.

Feedback

It is recommended to remove any interface declaration file that might be left over from a previous session by doing the following before the first analyze command.

```
file delete -force [get_app_var hdlin_persistent_sv_interfaces_filename]
```

See Also

- hdlin_persistent_sv_interfaces_filename

---

hdlin_port_dimension_mismatch_error

Vector or memory size mismatch on port results in an error instead of a warning.

Data Types

Boolean

Default true

Description

The *hdlin_port_dimension_mismatch_error* application options allows the user to get an error instead of a warning when a vector or memory size mismatch on port is detected, e.g.

```
module m1 (a);

   input [3:0] a;

   wire [5:0] a;

   endmodule
```

See Also

- analyze
- VER-151
- VER-152

---

hdlin_preserve_sequential

Controls whether the *elaborate* and *read* commands retain unloaded sequential cells in the design.

Data Types

string

Default none

Description

This variable instructs the *elaborate* command or the *read* command to retain unloaded sequential cells in the design. The default value is *none*. The following values are allowed:

none or false

No unloaded sequential cells are preserved.

all or true

All unloaded sequential cells are preserved, excluding unloaded sequential cells that are used solely as loop variables.

all+loop_variables or true+loop_variables

All unloaded sequential cells are preserved, including unloaded sequential cells that are used solely as loop variables.

ff

Only flip-flop cells are preserved, excluding unloaded sequential cells that are used solely as loop variables.

ff+loop_variables

Only flip-flop cells are preserved, including unloaded sequential cells that are used solely as loop variables.

latch

Only unloaded latch cells are preserved, excluding unloaded sequential cells that are used solely as loop variables.

latch+loop_variables

Only unloaded latch cells are preserved, including unloaded sequential cells that are used solely as loop variables.

When elaborating an RTL-level design description, HDL Compiler (Presto) infers a sequential cell (a latch or flip-flop) for any Verilog *reg* or VHDL *variable* objects that are conditionally assigned or are assigned under a clock edge. However, if the design never reads the values of these objects or never uses the values to compute any output of the design, the HDL Compiler (Presto) tool, by default, does not retain sequential cells for these objects. These sequential cells have no loads or there is no path from them to the outputs, and so the tool does not need them to implement the design.

You may want to preserve unnecessary sequential cells, however, to improve observability of the design. The *hdlin_preserve_sequential* variable controls which unloaded sequential cells remain in the design.

Feedback

The *+loop_variables* option affects which sequential cells are inferred when the value of *hdlin_preserve_sequential* is not set as *false* or *none*, and the *+loop_variables* option is not specified, the tool treats index variables as special and no sequential cells are inferred.

If you do not want variables treated in this special way, specify *+loop_variables* for the variable values.

In the following example module, the bits of *fail* would ordinarily not exist as sequential cells after elaboration of the design. However, note the following:

- If you set the value of *hdlin_preserve_sequential* as *all*, *true*, or *ff*, the tool infers sequential cells for *fail*, and since the *+loop_variables* option is not specified, the tool treats *i* in the example as special, because *i* is used only as a loop variable. Therefore, the tool does not infer sequential cells for *i*.
- If you set the value of *hdlin_preserve_sequential* as *all+loop_variables*, *true +loop_variables*, or *ff+loop_variables*, the tool infers sequential cells for *fail*, and since the *+loop_variables* option is specified, the tool treats *i* the same way as it treats all other variables and assigns *i* to sequential cells.
- If you set the value of *hdlin_preserve_sequential* as *none* (the default), *false*, or *latch*, the tool does not assign either *fail* or *i* to sequential cells in the design.

  Example 1:

```
module test (clk,rst,error);
    parameter N = 8;
    input clk,rst,error;
    reg   [N-1:0] fail;
    integer i, j;

    always @(posedge clk or posedge rst)
      if (rst)
        fail <= {N-1{1'b0}};
      else
        for ( i = 0; i < N; i = i + 1 )
          fail[i] <= fail[i] | error;
  endmodule // test
```

Table 1 shows how the variable settings affect the sequential cell inferring for *fail* and *i* in Example 1.

Asterisks (*) indicate that Presto infers the sequential cells.

```
Table 1 Variables and Sequential Cell Inferring Matrix for module test

-----------------------------------------+---------------+-------------
---
hdlin_preserve_sequential                |   all/true/ff  |
 none/false/latch
```

Feedback

```
----------------------------------------+--------+-------+--------+-----
---
option "+loop_variables" is specified    |  true  | false |  true  |
 false
----------------------------------------+--------+-------+--------+-----
---
reg [7:0]  fail                           *         *
reg [31:0] i                              *
-------------------------------------------------------------------------
---
```

Important: The process of elaboration might preserve unloaded sequential cells, but the cells are deleted by the *compile* command (by default). To preserve the cells throughout the compile process, set the value of the *compile_delete_unloaded_sequential_cells* variable as *false*.

If there are other variables in your design that cause sequential elements to be inferred, but that you do not want in the final design, specify the preservation of sequential cells on a variable-by-variable basis, using the Verilog *preserve_sequential* pragma in a declaration.

For more information on the *preserve_sequential* pragma, see the *HDL Compiler for Verilog User Guide* or the *HDL Compiler for VHDL User Guide*.

To determine the current value of this variable, use the *printvar hdlin_preserve_sequential* command. For a list of HDL variables and their current values, use *printvar_variable_group hdl*.

See Also

- elaborate
- read
- compile_delete_unloaded_sequential_cells

---

hdlin_presto_cell_name_prefix

Sets the internal cell name prefix for Presto HDL Compiler.

Data Types

string

Default C

Group

hdl_variables

Feedback

Description

This variable sets the internal cell name prefix for HDL Compiler. Use the *hdlin_presto_cell_name_prefix* variable to specify the cell name prefix. If you do not specify the variable, HDL Compiler generates the cell name with a "C" prefix.

The cell name prefix is only applied to cells whose names are not determined by another HDL Compiler cell naming convention.

The default value for the *hdlin_presto_cell_name_prefix* variable is *C*.

To determine the current value of this variable, use the *printvar hdelin_prest_cell_name_prefix* command. For a list of HDL variables and their current values, use *printvar_variable_group hdl*.

---

hdlin_presto_net_name_prefix

Sets the internal net name prefix for Presto HDL Compiler.

Data Types

string

Default N

Group

hdl_variables

Description

This variable sets the internal net name prefix. The net name generated by Synopsys DB always has the prefix "n". The net name generated by Presto has the prefix "N" by default. Use this variable to set the Presto net name prefix to avoid inconsistent net names. The default value for the *hdlin_presto_net_name_prefix* variable is *N*.

To determine the current value of this variable, use the *printvar hdlin_presto_net_name_prefix* command. For a list of HDL variables and their current values, use *printvar_variable_group hdl*.

---

hdlin_prohibit_nontri_multiple_drivers

Controls whether the HDL Compiler tool issues an error or warning message when it finds multiple drivers of a net.

Data Types

Boolean

Feedback

Default true

Description

This variable controls whether the HDL Compiler tool issues an error or warning message when it finds multiple drivers of a net. The variable supports Verilog and SystemVerilog input formats.

Multiple drivers occur when a reg variable is driven by more than one always block or a wire variable is driven by more than one continuous assignment or input port and all of the drivers are not tristates.

Before the 2001.08 release, the tool does not issue an error message for multiple-driver nets. After version 2001.08, the tool issues an error message by default. You can convert this ELAB-366 error message to an ELAB-365 warning by setting this variable to *false*. However, this setting is provided primarily for backward compatibility and is not recommended. You should inspect such warnings carefully. Setting the *hdlin_prohibit_nontri_multiple_drivers* variable to *false* can result in invalid designs.

In simulation, if a reg variable is driven by more than one always block, the definition depends on which block executed most recently. In synthesis, because all blocks are concurrently executing at all times, this behavior is not possible and invalid designs can result if the drivers are shorted together. Therefore, the tool issues an error message if any bits of any variables are driven by more than one process.

The Verilog standard prohibits a multiple-driven wire variable, although some simulators permit this behavior. The tool issues an ELAB-366 error message in this situation. If you want multiple-driven wire variables, set the *hdlin_prohibit_nontri_multiple_drivers* variable to *false*. Note that this variable setting might cause invalid designs.

Multiple drivers are permitted on tri nets, although a warning is issued if all the drivers are not tristate devices.

To determine the current value of this variable, use the *printvar hdlin_prohibit_nontri_multiple_drivers* command. For a list of all HDL variables and their current values, use the *print_variable_group hdl* command.

---

hdlin_report_info

Enables runtime and memory usage info to be displayed when elaborating modules.

Data Types

Boolean

Default true

Feedback

Description

The *hdlin_report_info* application option allows the user to get runtime and memory usage information when elaborating a module. The message is shown when the runtime or memory usage goes over a threshold set by *hdlin_report_time* and *hdlin_report_mem* respectively.

Example of the message:

```
Module: top, Elapsed Time: 00:35:01, CPU Time: 00:37:12, Total Mem: 5.09
 GB, Mem: 3.29 GB, Time: Mon May 12 16:45:13 2023
```

See Also

- elaborate
- hdlin_report_time
- hdlin_report_mem

---

hdlin_report_mem

Set minimum memory usage increase in MB when elaborating a module to show runtime and memory usage information.

Data Types

integer

Default 100

Description

The *hdlin_report_mem* application option allows the user to set the minimum memory usage increase in MB when elaborating a module to get runtime and memory usage information.

Example of the message:

```
set_app_var hdlin_report_mem 3000
Module: top, Elapsed Time: 00:35:01, CPU Time: 00:37:12, Total Mem: 5.09
 GB, Mem: 3.29 GB, Time: Mon May 12 16:45:13 2023
```

See Also

- elaborate
- hdlin_report_info
- hdlin_report_time

Feedback

hdlin_report_sequential_pruning

Controls reporting about removal of tentatively inferred sequentials.

Data Types

Boolean

Default false

Description

The *hdlin_report_sequential_pruning* variable lets you control whether HDL Compiler should issue a message when it detects that a sequential element that has already been tenatively inferred, but that is not actually required for proper functioning of the design being elaborated, is removed.

An RTL-level design description in Verilog or VHDL may contain variables for which a sequential cell would ordinarily be inferred (either because they are assigned under a clock edge, or because they are conditionally assigned) but whose values are never read, or whose values are not used in computing any output of the design. Because their values are not used (that is, they have no loads, or no path to the outputs), by default HDL Compiler will attempt not to infer sequential cells for these variables.

But if it does infer them, it will then try to prune them before the end of elaborate.

In some circumstances designers wish to retain these sequential cells. Unloaded sequential cells can be inferred by two methods. First, if hdlin_preserve_sequential is "all", then a sequential cell will be inferred for any variable that is conditionally assigned or assigned under a clock, regardless of whether its value is used or not. Second, if hdlin_preserve_sequential is not set, but some variables are marked with the "preserve_sequential" pragma, only those variables will receive sequential cells even if they have no loads.

See Also

- hdlin_preserve_sequential
- elaborate

hdlin_report_time

Set minimum elapsed time in minutes when elaborating a module to show runtime and memory usage information.

Data Types

integer

Feedback

Default 1

Description

The *hdlin_report_time* application option allows the user to set the minimum elapsed time in minutes when elaborating a module to get runtime and memory usage information.

Example of the message:

```
set_app_var hdlin_report_time 30
Module: top, Elapsed Time: 00:35:01, CPU Time: 00:37:12, Total Mem: 5.09
 GB, Mem: 3.29 GB, Time: Mon May 12 16:45:13 2023
```

See Also

- elaborate
- hdlin_report_info
- hdlin_report_mem

---

hdlin_reporting_level

Determines which information Presto HDL Compiler prints in the report.

Data Types

string

Default basic

Description

This variable controls the amount of output information to be included in the Presto elaboration report.

The following information is under control:

- *floating_net_to_ground* prints the report for floating net connects to ground. This variable is best used in conjunction with the *set hdlin_keep_signal_name user* command and is not guaranteed to report all nets. Use the *check_design* command for detecting unconnected pins and ports.
- *fsm* prints the report for inferred state variables.
- *inferred_modules* prints the report for inferred sequential elements.
- *mux_op* prints the report for MUX_OPs.
- *syn_cell* prints the report for synthetic cells.
- *tri_state* prints the report for inferred tristate elements.

Feedback

The *hdlin_reporting_level* variable can be set to 4 base settings: *none*, *basic*, *comprehensive*, and *verbose*, as shown in the following table:

Table 1 Base Settings

Information included in report	none	basic	comprehensive	verbose
floating_net_to_ground	false	false	true	true
fsm	false	false	true	true
inferred_modules	false	true	true	verbose
mux_op	false	true	true	true
syn_cell	false	false	true	true
tri_state	false	true	true	true

In addition to the base settings above, you can also modify the base settings to have fine grain control of individual reports by either adding (+) or subtracting (-) specific report(s) from the base setting with the following keywords:

```
floating_net_to_ground
fsm
syn_cell
mux_op
inferred_modules
tri_state
```

Examples

The following example uses *comprehensive-fsm*:

```
set hdlin_reporting_level comprehensive-fsm
```

The generated report shows the following settings:

```
floating_net_to_ground  true
fsm                     false
inferred_modules        true
mux_op                  true
syn_cell                true
tri_state               true
```

The following example uses *verbose-mux_op-tri_state*:

```
set hdlin_reporting_level verbose-mux_op-tri_state
```

The generated report shows the following settings:

```
floating_net_to_ground  true
fsm                     true
inferred_modules        verbose
mux_op                  false
syn_cell                true
tri_state               false
```

The following example shows two commands that generate equivalent reports:

```
set hdlin_reporting_level basic+floating_net_to_ground+syn_cell+fsm

set hdlin_reporting_level comprehensive
```

To determine the current value of this variable, use the *printvar hdlin_reporting_level* command. For a list of HDL variables and their current values, use *printvar_variable_group hdl*.

---

hdlin_shorten_long_module_name

Controls whether the Presto HDL Compiler compresses long names for elaborated modules.

Data Types

Boolean

Default true

Group

hdl_variables

Description

This variable controls whether the Presto HDL Compiler compresses long names for elaborated modules.

When the value of this variable is *true*, if the name of an elaborated module is longer than the value of *hdlin_module_name_limit* (default 256), then a compressed name is used that is easier for downstream tools to handle. The initial part of the name is still recognizable, but the tail of the name is replaced with numbers.

When the value is *false* (the default), the names of elaborated modules are uncompressed, even if they exceed the length specified by *hdlin_module_name_limit*.

To determine the current value of this variable, use the *printvar hdlin_module_name_limit* command. For a list of HDL variables and their current values, use *printvar_variable_group hdl*.

Feedback

See Also

- hdlin_module_name_limit

---

hdlin_strict_template_naming_style

Force user defined template_naming/parameter_styles for numeric parameters

Data Types

Boolean

Default false

Description

Force user defined template_naming/parameter_styles for numeric parameters. This is particularly useful for integer parameters expresed in binary, octal or hex formats.

Example

Following example is valid if: hdlin_strict_template_naming_style --> %s_%p (the default value). module shift_reg #(parameter WIDTH = 8, DEPTH = 8, NONINT = "abc") (input clk, rst, en, input [WIDTH-1:0] shift_in, output [WIDTH-1:0] shift_out); # Module instantiation --> modulename shift_reg #(.WIDTH(7), .DEPTH(8)) exE --> shift_reg_WIDTH7_DEPTH8 shift_reg #(.DEPTH(4'b0100), .NONINT("cde")) exC --> shift_reg_DEPTH4_NONINTcde shift_reg #(.WIDTH(4'h0001), .DEPTH(9)) exF --> shift_reg_WIDTH1_DEPTH9 shift_reg #(.WIDTH(4'o0001), .DEPTH(10), .NONINT("hij")) exG --> shift_reg_WIDTH1_DEPTH10_NONINThij

---

hdlin_strict_verilog_reader

Controls whether the Verilog Netlist Reader enforces strict IEEE-1364 language specification compliance.

Data Types

Boolean

Default false

Group

hdl_variables

Description

This variable controls whether the Verilog Netlist Reader errors or not on illegal verilog, like repeated identifiers.

When the value of this variable is *true*, constructs usually accepted by the verilog reader, such as instantiations with the same name as a wire, will trigger an error.

When the value is *false* (the default), no error is issued.

---

hdlin_subprogram_default_values

Determines which value the compiler will use as the default value for variables, 'LEFT of its type or 0s.

Data Types

Boolean

Default false

Group

hdl_variables

Description

This variable determines which value the compiler will use as the default value for variables, 'LEFT of its type or 0s.

When this variable is set to *true*, 'LEFT of the type of variable is used as its default value. If you set this variable to *false* (the default), 0s are used.

---

hdlin_sv_blackbox_modules

Specify one or more SystemVerilog modules to be ignored during design read.

Data Types

string

Default ""

Description

The *hdlin_sv_blackbox_modules* allows the user to specify one or more SystemVerilog modules to be ignored during design read.

The modules are specified by setting the variable *hdlin_sv_blackbox_modules* to a list of modules, e.g.

prompt> set hdlin_sv_blackbox_modules "mod1 mod2 ..."

Here mod1, mod2 are names of modules as coded in the RTL.

Feedback

The warning message (VER-746) is generated during *read_sverilog* or *analyze -f sverilog* command. An example of the message is shown below:

Warning: mod1.v:2: The declaration of module 'mod1' is being ignored, because the module name is in hdlin_sv_blackbox_modules. (VER-746)

If the tool tries to link the modules (during *elaborateor link commands), LINK-35 message is generated:*

Warning: All references to module 'mod1' are ignored and treated as black boxes. (LINK-35)

The valid module names are those coded in the RTL. The design names, e.g. those reported by *list_designs, may not work with this feature.*

No messages are generated for specified names that do not match valid module names.

See Also

- read_sverilog
- analyze
- elaborate
- link
- list_designs

---

hdlin_sv_enable_rtl_attributes

Enable SystemVerilog user attributes.

Data Types

boolean

Default FALSE

Description

The *hdlin_sv_enable_rtl_attributes* SystemVerilog allows users to specify their own attributes on cell, port, pin, and module. For example:

```
module top (
    (* user_attr = "true" *) input TOPIN,
    (* user_attr = "top port" *) output TOPOUT);
   (* my_attr="P" *) BOT U_BOT ((* Pin_Attr = 0
 *) .BOTIN(TOPIN), .BOTOUT(TOPOUT));
  endmodule
```

Feedback

```
dc_shell> report_attributes [get_ports *]
Design        Object          Type        Attribute Name        Value
-------------------------------------------------------------------------
-------
top           TOPIN           string      user_attr             true
top           TOPOUT          string      user_attr             top port

dc_shell> report_attributes [get_cells *]
Design        Object          Type        Attribute Name        Value
-------------------------------------------------------------------------
-------
top           U_BOT           string      my_attr               P
```

See Also

- analyze
- elaborate
- link

---

hdlin_sv_enforce_standalone_generate_blocks

Enforce that standalone generate blocks will be banned in SystemVerilog and Verilog-2005

Data Types

Boolean

Default true

Group

hdl_variables

Description

Enforce that generate blocks have either a loop or conditional construct according to the SystemVerilog and Verilog-2005 LRMs, i.e. the standalone generate block will be banned. Setting this variable to false will reduce the severity of using a standalone generate from an error (VER-946) to a warning (VER-945).

See Also

- hdlin_vrlg_std
- hdlin_sverilog_std

Feedback

- VER-945
- VER-946

---

hdlin_sv_interface_only_modules

Enable HDL Compiler to read SystemVerilog designs as interface only

Data Types

string

Default ""

Description

This option lists System Verilog designs as interface only. HDL Compiler parses the module interface of the listed designs, skipping the module content, and creates a black box for each module.

During elaboration, HDL compiler issues a warning message that the module content is discarded and ignored.

Here is an example:

prompt> set hdlin_sv_interface_only_modules {my_module1 my_module2} prompt> analyze -f sverilog top.sv

Warning: ./rtl/top.sv:21: The body of module 'my_module1' is being discarded, because the module name is in hdlin_sv_interface_only_modules. (VER-747)

---

hdlin_sv_packages

Specifies how SystemVerilog packages should be analyzed.

Data Types

string

Default dont_chain

Description

Specifies which semantics the *analyze* or *read_file* command should apply when a *-format sverilog* source file declares a package. The setting affects the analyze step of SystemVerilog package declarations or references; it has no effect during *elaborate*. The allowed values for *hdlin_sv_packages* are *dont_chain* (the default) and *chain*.

The default behavior is as described in Clause 26 of IEEE Std 1800-2012, the standard for SystemVerilog. Overriding the default to *chain* changes how an *import* statement treats names imported into the topmost (global) scope of a package_declaration:

- The default *dont_chain* setting enforces the IEEE standard and prevents imported names from being re-exported to clients of the package being declared.
- The *chain* setting instead re-exports names that are imported into the global scope of a package; they may all be imported by the intermediate package's clients. An imported name and its definition that are re-exported (or chained) will not collide or interfere with copies of themselves in those cases where several intermediate packages redistribute content they acquired from a common source package (provided they all acquire it from a compatible analyzed version of the same source file).

Example: By default, output dont_chain will get 1 and output chain will get 0, but if the application option is overridden to *chain*, then output dont_chain will get 0 and output chain will get 1.

```
package A; localparam N = 44; endpackage
package B; localparam N = 77; endpackage
package C; import B::N; endpackage
import A::*;
module VER934 import C::*; (output dont_chain, chain);
  assign dont_chain = (N == 44);
  assign chain = (N == 77);
endmodule
```

The analyzed result, a file named *package_identifier*.pvk, always contains a full copy of all imported content. A package's .pvk file can stand alone; it does not require its clients to access the .pvk files that supplied its imported ingredients (unlike source-level file inclusion).

The chaining issue only concerns whether imported names become explicitly visible to an intermediate package's clients as do objects explicitly declared at the outermost level of the intermediate package. Because a wildcard "import intermediate_pkg::*;" encumbers all of the exportable names found in "intermediate_pkg", the selection of *chain* or *dont_chain* can alter the outcome of name resolutions when several packages are combined in a downstream client.

The setting of *hdlin_sv_packages* at the time a package is analyzed is compiled into a visibility property on the imported, global names in the resulting .pvk file. This is an independent property that can be different at each level of a supply chain; it is not an inherited property of the name itself. A VER-934 informational message always indicates how this property is being set for those names where it might eventually matter.

To determine the current value of this variable, use the *report_app_var hdlin_sv_packages* command. For a list of HDL variables and their current values, use *print_variable_group hdl*.

Feedback

See Also

- analyze
- read_file
- VER-20
- VER-21
- VER-934

---

hdlin_sv_tokens

Specifies whether a tokens file should be written out during the analysis of SystemVerilog designs.

Data Types

Boolean

Default false

Description

When *hdlin_sv_tokens* is turned on, the lexical tokens that are sent to the parser during analysis of a SystemVerilog design (such as with an *analyze -f sverilog* or a *read_sverilog* command) will also be saved to an output file in the work directory.

When using complex macros or nested conditional compilation directives, it might help to see what the actual RTL will be after preprocessing, for example, to understand a syntax error.

When the value of *hdlin_sv_tokens* is overridden and set to *true*, an expanded version of the source files is written to an output file *tokens.n{n}.sv*, instead of being sent to the parser. In the output file, conditional compilation directives will already have been taken into account and all macro invocations will already have been expanded. The first tokens file written out in a particular work directory will be *tokens.1.sv*, the next *tokens.2.sv*, the next *tokens.3.sv*, and so on.

Although the output file is legal SystemVerilog, it is formatted in a way that is intended to be conveniently human readable, by using standard line directives to indicate the original source files and line numbers. This makes it relatively easy to track down the source of an error using the file name and line number in the error message. For example, if the error message refers to line 321 of the file *"my_file.sv"*, then look for *line 321 "my_file.sv"* in the output file.

The only comments preserved are those that are sent to the parser, because they are in more than mere comments, such as synthesis pragmas and embedded scripts.

Feedback

If any of the source files are encrypted, then the output file is not created.

See Also

- analyze
- read_sverilog

---

hdlin_sv_union_member_naming

Controls the naming styles for elements associated with the union data type in SystemVerilog.

Data Types

Boolean

Default false

Group

hdl_variables

Description

By default, the tool uses simple names for elements inferred from unions in SystemVerilog. Setting this variable to *true* enables the tool to use the name of the first union member as a reference for the port, net, and cell names associated with the union data type. For example,

```
1 typedef union packed {
  2   byte        field1;
  3   logic [7:0] field2;
  4 } packet;
  5
  6 module test (input packet p1, output packet p2, input clk);
  7   always_ff @ (posedge clk)
  8     p2.field1 = p1.field2;
  9 endmodule
```

When you set this variable to *false*, the following names are inferred:

```
Ports and nets: p1[7], p1[6], ..., p1[0]
    Cells: p2_reg[7], p2_reg[6], ..., p2_reg[0]
```

When you set this variable to *true*, the following names are inferred:

```
Ports and nets: p1[field1][7], p1[field1][6], ..., p1[field1][0]
    Cells: p2_reg[field1][7], p2_reg[field1][6], ..., p2_reg[field1][0]
```

Feedback

When you use the *read_saif* command to read in a SAIF file generated from a FSDB, setting this variable to *true* might increase the acceptance of the annotated switching activity.

hdlin_sv_use_search_path_for_include

Look for a file in search path if they are not found in the given path in a verilog *include* statement.

Data Types

Boolean

Default false

Group

hdl_variables

Description

Extend the behavior of the *includestatement in Verilog, looking for a file in the directories in the 'search path' variable if the file is not found in the provided path. By default, if a file is given with a relative or absolute path in a include statement, it is searched for only on the provided path, failing if it is not found there.*

See Also

- analyze

hdlin_sverilog_std

Controls whether HDLC SystemVerilog enforces SystemVerilog 2005, 2009, 2012, or 2017.

Data Types

integer

Default 2017

Group

hdl_variables

Description

This variable controls whether HDLC SystemVerilog is to enforce SystemVerilog 2005 or SystemVerilog 2009 or SystemVerilog 2012 or SystemVerilog 2017.

Feedback

- If the variable is set to 2017 (the default), SystemVerilog 2017 is enforced. There is no Verilog 2017.
- If the variable is set to 2012, SystemVerilog 2012 is enforced. There is no Verilog 2012.
- If the variable is set to 2009, SystemVerilog 2009 is enforced. There is no Verilog 2009.
- If the variable is set to 2005, SystemVerilog 2005 is enforced.

The SystemVerilog 2009 standard merged two previous standards, Verilog 2005 and SystemVerilog 2005, which defined extensions to it. Those two standards were designed to be used as one language, so merging the base Verilog language and the SystemVerilog extensions into a single standard provides all information regarding syntax and semantics in a single document. Additionally, there are many extensions beyond SystemVerilog 2005 in SystemVerilog 2009.

The SystemVerilog 2012 standard adds extensions beyond SystemVerilog 2009.

The SystemVerilog 2017 standard clarifies some details of SystemVerilog 2012 and reserves no additional keywords.

See Also

- hdlin_vrlg_std

---

hdlin_tic_tic_discards_whitespace

Data Types

Boolean

Default true

Description

As described in section 22.5.1 of IEEE Std 1800-2017, a `` (double backtick) in a SystemVerilog macro definition "delimits lexical tokens without introducing white space, allowing identifiers to be constructed from arguments."

Enabling this application variable (default is *true*) brings the tool into alignment with simulators on a related topic that is not clarified in the standard, because any whitespace surrounding macro arguments will be removed unless that whitespace closes an escaped identifier.

For example, unless this application variable is enabled, analyzing the following RTL:

```
module top(output test_bit);
  `define M(arg) wire \_``arg``_ [2];
  `M(   t@st    )
  assign test_bit = \_t@st_ [0] & \_t@st_ [1];
```

Feedback

```
  assign \_t@st_ = '{default:1'b1};
  initial assert final (test_bit);
endmodule
```

results in the following message:

```
Error:  ./top.sv:3: Syntax error at or near token '_'
        in macro "M"
        called from file "./top.sv" (line 3). (VER-294)
```

---

hdlin_unified_rtl_read

Enable Synopsys unified frontend

Data Types

Boolean

Default false

Description

This is the main control variable that decides which flow analyze/elaborate commands will use.

When *hdlin_unified_rtl_readis true, analyze/elaborate commands will use Synopsys unified frontend for reading the RTL files.*

When *hdlin_unified_rtl_readis false (the default), analyze/elaborate commands will use Legacy frontend for reading the RTL files.*

To determine the current value of this variable, use the *printvar hdlin_unified_rtl_read* command. For a list of HDL variables and their current values, use *print_variable_group hdl*.

See Also

- hdlin_vcs_home
- analyze
- elaborate
- read_verilog
- read_file
- set_svf
- set_verification_top

Feedback

hdlin_upcase_names

Controls whether identifiers in the Verilog source code are converted to uppercase letters or left in their original case.

Data Types

Boolean

Default false

Description

This variable controls whether identifiers in the Verilog source code are converted to uppercase letters or left in their original case.

The default setting of this variable is *false*, which means that all names are left the way they are.

Setting the variable to *true* causes conversion of all identifiers in the Verilog code (variables, ports, and so on) to uppercase letters.

hdlin_v2005_replication_semantics

Controls the treatment of zero replication constants in Verilog-1995 and Verilog-2001.

Data Types

Boolean

Default true

Group

hdl_variables

Description

Until Verilog-2005 it was non-standard for the first operand in a replication operation to be zero, but, instead of issuing an error, HDL Compiler would implement {0{expression}} as 1'b0, regardless of the size or value of the second operand. By default, the tool now implements a zero replication constant according to the rules of Verilog-2005, even if the Verilog standard is overridden back to 1995 or 2001, in which case overriding this variable to *false* restores the older behavior. Overriding this variable has no effect when the standard is 2005. Because the older style of implementing a non-standard zero replication constant is likely to mismatch with simulation, an ELAB-364 warning is issued when it happens.

For example, in Verilog-2001 by default

Feedback

```
1 module test(input [1:0] in, output [3:0] out);
  2   assign out = { {2{in}}, {0{in}} };
  3 endmodule
```

is implemented the same as

```
1 module test(input [1:0] in, output [3:0] out);
  2   assign out = {2{in}};
  3 endmodule
```

but when the variable is overridden to *false* it is implemented the same as

```
1 module test(input [1:0] in, output [3:0] out);
  2   assign out = { in[0], in, 1'b0 };
  3 endmodule
```

and an ELAB-364 warning is issued.

See Also

- hdlin_vrlg_std

---

hdlin_vcs_home

VCS_HOME to take vcs binaries, when using unified frontend flow

Data Types

String

Default ""

Description

Users must set this variable to the path of a compatible version of VCS installation that tool should use.

To determine the current value of this variable, use the *printvar hdlin_vcs_home* command. For a list of HDL variables and their current values, use *print_variable_group hdl*.

See Also

- hdlin_unified_rtl_read
- analyze
- elaborate
- read_verilog

Feedback

- read_file
- set_svf
- set_verification_top

hdlin_verification_priority

Instructs the tool to prioritize formal verification over QoR while reading the RTL files.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Group

hdl_variables

Description

Setting this variable to *true* adjusts optimization inside of the tool to prioritize formal verification compatibility over QoR. This variable only affects optimizations done during the reading and elaboration of RTL files. To control optimization during later stages, see the man page for the *set_verification_priority* command.

Set this variable in response to a hard verification from your formal verification tool.

See Also

- set_verification_priority

hdlin_vhdl93_concat

Controls the concatenation behavior the tool uses to conform to the VHDL '93 Standard or the VHDL '87 Standard.

Data Types

Boolean

Default true

Feedback

Group

hdl_variables

Description

The *hdlin_vhdl93_concat* variable controls the concatenation behavior the Presto VHDL Compiler uses to conform to the VHDL '93 Standard or the VHDL '87 Standard. If you set the value of this variable as *true* (the default), the tool follows the VHDL '93 Standard. If you set the value as *false*, the tool follows the VHDL '87 Standard.

---

hdlin_vhdl_mixed_language_instantiation

Controls if mixed language instantiations are to be used in VHDL.

Data Types

Boolean

Default false

Description

This variable controls if mixed language instantiations are to be used in VHDL.

When you set the *hdlin_vhdl_mixed_language_instantiation* variable to true, HDL Compiler allows VHDL direct entity instantiations of a lower-level design that is not VHDL. Furthermore, if VHDL configurations are enabled, you can configure (standalone or architecture embedded) a lower-level design that is not VHDL to be used for a VHDL component; that is, you can configure the components inside this lower-level design.

All parameters have to be mapped explicitly in the generic mappings, though one or more are set to the default. Additionally, you cannot map literals in the generic mappings; that is, literal constants, such as 11, should be replaced with local constants like my_bit_vector_constant.

To see the current setting of this variable, use the *printvar hdlin_vhdl_mixed_language_instantiation* command. For a list of HDL variables and their current settings, use the *print_variable_group hdl* command.

---

hdlin_vhdl_preserve_case

Controls if VHDL record fields preserve its case after HDLC.

Data Types

Boolean

Default false

Feedback

Description

This variable only affects to the case setting of the field naming on VHDL record type names, it does not impact other VHDL names. By default the field switch to uppercase after HDLC.

When *hdlin_vhdl_preserve_case* is set to *true* the fields will preserve its original case after HDLC. For example, take the following record:

```
type myType is record
  Reg_oNe          : std_logic;
end record;
```

If an output register is defined as myType It will be named as Out1_reg[REG_ONE] by default. With *hdlin_vhdl_preserve_case* set to *true* it will be named as Out1_reg[Reg_oNe].

See Also

- hdlin_enable_upf_compatible_naming
- hdlin_enable_hier_naming
- hdlin_field_naming_style

---

hdlin_vhdl_std

Controls whether HDL Compiler follows the VHDL 2008 standard, the 1993 standard, or the 1987 standard.

Data Types

integer

Default 2008

Description

This variable determines the standard that the HDL Compiler is to follow in your design.

- If this variable is set to 2008 (the default), the compiler uses the VHDL 2008 standard.
- If this variable is set to 1993, the compiler uses the VHDL 1993 standard.
- If this variable is set to 1987, the compiler uses the VHDL 1987 standard.

See Also

- hdlin_sverilog_std
- hdlin_vrlg_std

Feedback

hdlin_vhdl_syntax_extensions

Enables VHDL language features that are currently outside of the supported synthesizable subset.

Data Types

Boolean

Default false

Description

This variable enables VHDL language features that are currently outside of the supported synthesizable subset. Use of these language constructs must be accompanied by thorough verification.

The following features are currently enabled:

- Deferred constant definition: a constant declaration not accompanied by an assignment expression.
- Arrays of base type Boolean: use Boolean as the range type of an array.
- Impure functions: a function specification contains the reserved word impure.

hdlin_vrlg_std

Controls whether Presto Verilog enforces Verilog 1995, Verilog 2001, or Verilog 2005.

Data Types

integer

Default 2005

Group

hdl_variables

Description

This variable controls whether Presto Verilog enforces Verilog 1995, Verilog 2001, or Verilog 2005.

- If this variable is set to the value of 2005 (the default), Verilog 2005 is enforced.
- If the variable is set to 2001, Verilog 2001 is enforced.
- If the variable is set to 1995, Verilog 1995 is enforced.

Feedback

See Also

- hdlin_sverilog_std

---

hdlin_while_loop_iterations

Places an upper bound on the number of times a loop is unrolled to prevent potential infinite loops.

Data Types

string

Default 4096

Description

This variable places an upper bound on the number of times a loop is unrolled to prevent potential infinite loops. Loop unrolling occurs until the loop terminates. If you know that your loop will execute more times than the limit allows and that your loop will terminate at some point, increase the value of this variable.

To determine the current value of this variable, use the *printvar hdlin_while_loop_iterations* command. For a list of HDL variables and their current values, use the *print_variable_group hdl* command.

---

hdlout_internal_busses

Controls the way in which the *write -format verilog* command and the *write -format vhdl* command write out internal bused nets by parsing the names of the nets.

Data Types

Boolean

Default false

Description

When the *hdlout_internal_busses* variable is set to the value of *true*, it controls the way in which the *write -format verilog* command and the *write -format vhdl* command write out internal bused nets by parsing the names of the nets.

When writing out VHDL files, be sure to set the *vhdlout_single_bit* variable and the *vhdlout_preserve_hierarchical_types* variable to "user" or "vector." Set the *bus_inference_style* variable and the *bus_naming_style* variable to the naming style. For more information, see the man pages for the *bus_inference_style* and *bus_naming_style* variables.

Feedback

When writing out Verilog files, set the *verilogout_single_bit* variable to *false* (the default). Set the *bus_inference_style* variable and the *bus_naming_style* variable to the naming style. For more information, see the man pages for the *bus_inference_style* and *bus_naming_style* variables.

To determine the current value of this variable, use the *printvar hdlout_internal_busses* command. For a list of all *io* variables and their current values, use the *print_variable_group io*.

See Also

- bus_inference_style
- bus_naming_style

---

hier_dont_trace_ungroup

Disables ungroup tracing set on the design with the *ungroup* command.

Data Types

Boolean

Default 0

Description

This variable disables ungroup tracing set on the design with the *ungroup* command. When *hier_dont_trace_ungroup* is set to *0* (the default), the *ungroup* command places on the design being ungrouped a string attribute that describes the ungroup operation. Other tools (for example, RTL Analyzer) can later use the attribute to recreate the ungroup operation and trace between the mapped and GTECH (generic) circuits.

Setting *hier_dont_trace_ungroup* to *1* disables ungroup tracing and can increase the efficiency of *ungroup* and other commands.

See Also

- ungroup

---

high_fanout_net_pin_capacitance

Specifies the pin capacitance to use for computing the loading of high-fanout nets.

Data Types

float

Feedback

Default 1

Description

This variable specifies the pin capacitance to use for computing the loading of high-fanout nets.

The tool computes the pin capacitance for high-fanout nets by multiplying the capacitance specified with the *high_fanout_net_pin_capacitance* variable times the high-fanout threshold.

For best results, you should specify a large value for the pin capacitance to cause violations on all constrained high-fanout nets. This forces the tool to replace the nets with buffer trees during compilation.

To determine the current value of this variable, use the *printvar high_fanout_net_pin_capacitance* variable. For a list of all timing variables and their current values, use the *print_variable_group timing* variable.

See Also

- high_fanout_net_threshold

---

high_fanout_net_threshold

Specifies the minimum number of loads for a net to be classified as a high-fanout net.

Data Types

integer

Default 1000

Description

This variable specifies the minimum number of loads for a net to be classified as a high-fanout net.

Delays and loads of high-fanout are computed using a simplified model assuming a fixed fanout number. The rationale behind this is that delays of high-fanout nets are expensive to compute but such nets are often unconstrained (as in the case of global reset nets, scan enable nets, and so on). Those high-fanout nets that actually are constrained should eventually be replaced by buffer trees. So detailed delay calculations on such nets are expensive and usually unnecessary.

Setting the threshold to 0 (or to a very large number) ensures that no nets will be treated as high-fanout nets. However, be aware that forcing fully accurate delay calculations on high-fanout can significantly increase compilation runtime in some cases.

The pin capacitance for high-fanout nets is computed by multiplying the capacitance specified by the *high_fanout_net_pin_capacitance* variable times the high-fanout threshold plus the number of net drivers. If there are delay annotations remaining from an earlier flow step, these annotations will be used instead of the high-fanout net model.

The simplified net delay model is used only when computing data delays. Propagated clock latencies are always computed using the full accuracy net delay model.

To determine the current value of this variable, use the *printvar high_fanout_net_threshold* command. For a list of all timing variables and their current values, use *print_variable_group timing*.

See Also

- high_fanout_net_pin_capacitance

---

hlo_resource_allocation

Sets the default resource sharing type to be used by the *compile* command, if the *resource_allocation* attribute is not set.

Data Types

string

Default constraint_driven in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable sets the default resource sharing type to be used by the *compile* command, if the *resource_allocation* attribute is not set. This variable has no effect in *compile_ultra*.

Allowed values are as follows:

constraint_driven (the default)

Directs *compile* to share resources so that the timing constraints are met, or not made worse, by sharing.

area_only

Directs *compile* to share operators without regard for timing constraints. All arithmetic expression trees are balanced.

Feedback

area_no_tree_balancing

Directs *compile* to share operators without regard to timing constraints. Arithmetic expression trees are not balanced.

none

Directs *compile* to do no resource sharing, so that each operation is implemented with separate circuitry.

true

Equivalent to *constraint_driven*; provided for backward compatibility with v2.0.

false

Equivalent to *none*; provided for backward compatibility with v2.0.

You can override the value of *hlo_resource_allocation* for the current design by using the *set_resource_allocation* command to set the *resource_allocation* attribute.

To determine the current value of this variable, use the *printvar hlo_resource_allocation* command. For a list of all HDL variables and their current values, use *print_variable_group hdl*.

See Also

- compile
- set_resource_allocation
- design_attributes
- synthetic_library

---

html_log_enable

Enables the HTML log generation in dc_shell.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Group

system_variables

Feedback

Description

This variable enables the HTML log generation in dc_shell, which then can be viewed in the file that the variable html_log_filename points to.

See Also

- html_log_filename

---

html_log_filename

Changes the file name of the starting HTML file.

Data Types

String

Default default.html in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Group

system_variables

Description

This variable changes the file name of the starting HTML file that is generated if the HTML log generation is enabled in dc_shell.

See Also

- html_log_enable

---

i

---

icc2_link_auto_fp_enable_multi_height_rows

When set, enables site row generation for pre-instantiated multi-height cells during ICC2 link auto-floorplan.

Data Types

Boolean

Feedback

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

true in Design Compiler NXT topographical mode

Description

When variable *icc2_link_auto_fp_enable_multi_height_rows* is set to *true* before *compile_ultra -spg*, and *set_auto_floorplan_constraints -site_def* is specified, tool checks for any multi-height cell in the design, and creates multi-height rows too during ICC2 Link based auto-floorplan.

The feature is not supported when *set_auto_floorplan_constraints* uses either of options *-row_pattern*, *-row_core_ratio* or *-flip_first_row*.

This feature in supported in Design Compiler NXT topographical NDM mode.

Examples

The following example shows how to enable and disable mutli-height row generation during ICC2 link based auto-floorplan.

```
prompt> set icc2_link_auto_fp_enable_multi_height_rows true
prompt> set icc2_link_auto_fp_enable_multi_height_rows false
```

See Also

- dcnxt_use_icc2_link
- set_auto_floorplan_constraints
- report_auto_floorplan_constraints

icc2_link_ddc_transfer

Enables the DDC based transfer from Design Compiler topographical to ICC2.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode and Design Compiler NXT topographical mode

Feedback

Description

Setting the *icc2_link_ddc_transfer* variable to *true* enables DDC based transfer through the ICC2Link from Design Compiler topographical to ICC2.

This variable works only in topographical mode.

To determine the current value of this variable, use *printvar icc2_link_ddc_transfer* command.

See Also

- set_icc2_options

---

icc2_link_enable_autofp_port_macro_placement

When set, enables ICC2 link based macro and port placement.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

true in Design Compiler NXT topographical mode

Description

When variable *icc2_link_enable_autofp_port_macro_placement* is set to *true* before *compile_ultra*, the tool invokes ICC2 link based port/macro placement, if needed. The ICC2 link based ports/macros placement is enabled if - the design contains unplaced ports and/or macros - first placer is native placer - one of the following commands are specified (1) *set_auto_floorplan_constraints* or (2) *dcnxt_use_icc2_link -auto_floorplan true*

This feature in supported in Design Compiler NXT topographical NDM mode.

Examples

The following example shows how to enable and disable ICC2 link based ports and/or macro placement:

```
prompt> set icc2_link_enable_autofp_port_macro_placement true
prompt> set icc2_link_enable_autofp_port_macro_placement false
```

Feedback

See Also

- dcnxt_use_icc2_link
- set_auto_floorplan_constraints
- report_auto_floorplan_constraints

---

icc2_link_enable_autofp_track_generation

When set, enables technology node aware IC Compiler II RM script based track generation.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

true in Design Compiler NXT topographical mode

Description

When variable *icc2_link_enable_autofp_track_generation* is set to *true* before *compile_ultra*, the tool invokes technology node aware track generation during *ICC2-Link* based Auto-floorplanning. However, if the tool generated tracks do not align with the given libraries, please contact your library provider for correct track generation script and pass it through command *set_auto_floorplan_constraints -track_script*.

This feature in supported in Design Compiler NXT topographical mode.

Examples

The following example shows how to enable and disable ICC2 link based track generation:

```
prompt> set icc2_link_enable_autofp_track_generation true
prompt> set icc2_link_enable_autofp_track_generation false
```

See Also

- dcnxt_use_icc2_link
- set_auto_floorplan_constraints
- report_auto_floorplan_constraints

Feedback

icc2_link_enable_reduced_log

Suppresses part of the ICC2 log from the main DCNXT compile log during ICC2 link.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

false in Design Compiler NXT topographical mode

Group

icc2

Description

During ICC2-Link, DCNXT will no longer echo non-critical parts of the ICC II log file. This allows the DC log file to be clean of ICC II link design transfer and setup messages. All Error and Warning messages will be printed in IC Compiler II run log file in work directory.

icc2_link_processes_inherit_parent_process_group

When set, any new processes spawned by ICC2 Link will inherit the parent process group.

Data Types

Boolean

Default false

Group

icc2

Description

During ICC-II Link, DC spawns processes which may not have a parent process. When this variable is set to true, the ICC-II Link processes will never be orphaned. Instead, these spawned process will inherit the parent process group.

ignore_clock_input_delay_for_skew

Controls how clock skew calculations handle the input delay on clocks.

Feedback

Data Types

Boolean

Default false

Description

This variable controls how clock skew calculations handle the input delay on clocks.

Normally, you should use the *set_input_delay* command only on data ports, and use the *set_clock_latency* command on clock ports. In cases where the *set_input_delay* command has been used on a clock port, the default behavior differs between PrimeTime and the synthesis timing engine used by Design Compiler and IC Compiler.

By default (*false*), the synthesis timing engine uses the input delay values set on the clocks with the *set_input_delay* command when computing the skew values.

When set to *true*, the synthesis timing engine ignores the input delay values set on the clocks with the *set_input_delay* command when computing the skew values of clocked registers, which is compatible with the default PrimeTime behavior.

Set this variable to *true* when correlating synthesis and Primetime results.

See Also

- set_input_delay
- set_clock_latency

---

ignore_tf_error

Sets the flag for the tool to ignore unrecognized attributes in the technology file.

Data Types

Boolean

Default false

Description

Setting the *ignore_tf_error* variable to *true* allows the tool to ignore unrecognized attributes when reading the technology file. Unrecognized attributes will still be reported using the TFCHK-009 error message. However, a TFCHK-096 informational message will be reported at the end of the technology file reading to indicate that these errors were ignored.

Feedback

Since the technology file contains many attributes that are required for correct functionality of the tool, set this variable only after it has been verified that all of the unrecognized attributes in the technology file are safe to ignore.

See Also

- create_mw_lib

---

in_gui_session

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

Data Types

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
- gui_start
- gui_stop

---

initial_target_library

Specifies the list of technology libraries of components to be used for the first part of leakage power optimization in *place_opt*.

Data Types

list

Default "" in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Feedback

Description

Leakage power can be optimized for multi-vth designs in two different flows in *place_opt*. One way is to use all of the target libraries throughout optimization. The other is to use a subset of target libraries in the first part of the optimization steps and all the libraries in the second part. The *initial_target_library* variable specifies the list of technology libraries of components to be used for the first part of leakage power optimization in *place_opt*.

---

insert_test_design_naming_style

Specifies how the *insert_dft* command names new designs created during the addition of test circuitry.

Data Types

string

Default %s_test_%d

Description

This variable specifies how the *insert_dft* command names new designs created during the addition of test circuitry. When *insert_dft* modifies a design by adding test circuitry, it creates the design with a new, unique name. The new name is derived from the original design name and the format specified by this variable.

This variable must contain only one %s (percent s) and %d (percent d) character sequence. The percent sign has special meaning in the formatting process. To use a percent sign in the design name, two are needed in the variable setting (%%).

When *insert_dft* generates a new design name, it replaces %s with the original design name and %d with an integer. The integer is one that ensures the new name is unique. A single percent sign is substituted for %%.

For example, if this variable is set to *%s_test_%d*, and the original design name is *my_design*, the new design name is *my_design_test_1*.

If the *insert_dft* command is repeated (for example, with a different test methodology), the new design name is *my_design_test_2*.

To determine the current value of this variable, use the *printvar insert_test_design_naming_style* command. For a list of *insert_dft* variables and their current values, use *print_variable_group insert_dft*.

See Also

- insert_dft

Feedback

l

lbo_cells_in_regions

Puts new cells at specific locations within a cluster.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

Location based optimization (LBO) puts new cells at specific locations within a cluster. When this variable is set to *true*, LBO converts the specific location into a preferred region for the cell, by putting X_BOUNDS and Y_BOUNDS attributes on the cell when it is written to the PDEF file.

The proper setting for this variable depends on the engineering change order (ECO) capabilities of the back-end tools being used. Ideally the back-end tool is able to support putting the new cells exactly where the *reoptimize_design* command wants them to go. If tools do not support that level of ECO, set this variable to *true* so that the PDEF file will at least contain regions into which the cells can be placed.

To determine the current value of this variable, use the *printvar lbo_cells_in_regions* command. For a list of all *links_to_layout* variables and their current values, use *print_variable_group links_to_layout*.

level_shifter_naming_prefix

Specifies a prefix for level shifter names.

Data Types

string

Default ""

Group

mv

Feedback

Description

When the *enable_special_level_shifter_naming* variable is set to *true*, level shifters that are automatically inserted by the *insert_level_shifters*, *compile* and other commands are specially named. The name follows the template <prefix> + <PD OR Design name> + "_LS" + #. The <prefix> is a user-specified prefix using the *level_shifter_naming_prefix* variable. The <PD OR Design Name> is the name of power domain where the level shifter is being added, or the design name if the power domain is not defined

See Also

- enable_special_level_shifter_naming

---

lib_cell_using_delay_from_ccs

Controls whether the tool uses information from CCS tables instead of NLDM library timing information for cases with mixed CCS and NLDM libraries.

Data Types

Boolean

Default true

Description

When this variable is set to *true* and the library contains both CCS timing and NLDM timing, the tool replaces the NLDM library timing with the timing information derived from CCS. When the variable is set to *false* and the library contains both CCS timing and NLDM timing, the tool uses the NLDM library timing from library NLDM data. This variable only works when it is set before library loading.

For the current value of this variable, use the *printvar lib_cell_using_delay_from_ccs* command.

See Also

- lib_pin_using_cap_from_ccs

---

lib_pin_using_cap_from_ccs

Controls whether the tool uses the pin caps derived from CCS in cases with mixed CCS and NLDM libraries.

Data Types

Boolean

Feedback

Default true

Description

When this variable is set to *true* and the library contains both CCS timing and NLDM timing, the tool uses the pin caps derived from CCS. When the variable is set to *false* and the library contains both CCS timing and NLDM timing, the tool uses the NLDM library pin cap. This variable only works when it is set before library loading.

For the current value of this variable, use the *printvar lib_pin_using_cap_from_ccs* command.

See Also

- lib_cell_using_delay_from_ccs

---

lib_use_thresholds_per_pin

Causes pin-specific trip-point values in the Synopsys library to override library default trip-point values.

Data Types

Boolean

Default true

Description

Setting this variable to *false* causes Synopsys Library default trip-point values to override values defined for each library pin in the Synopsys library.

This variable is provided for backward compatibility. This variable allows you to use library defaults for all library pins instead of pin specific trip_point values.

The following variables are affected:

```
lib_thresholds_per_lib
rc_input_threshold_pct_rise
rc_input_threshold_pct_fall
rc_output_threshold_pct_rise
rc_output_threshold_pct_fall
rc_slew_derate_from_library
rc_slew_lower_threshold_pct_fall
rc_slew_lower_threshold_pct_rise
rc_slew_upper_threshold_pct_fall
rc_slew_upper_threshold_pct_rise
```

Feedback

To determine the current value of this variable, use the *printvar lib_use_thresholds_per_pin* command. For a list of all timing variables and their current values, use *print_variable_group timing*.

See Also

- rc_input_threshold_pct_fall
- rc_input_threshold_pct_rise
- rc_output_threshold_pct_fall
- rc_output_threshold_pct_rise
- rc_slew_derate_from_library
- rc_slew_lower_threshold_pct_fall
- rc_slew_lower_threshold_pct_rise
- rc_slew_upper_threshold_pct_fall
- rc_slew_upper_threshold_pct_rise

---

libgen_max_differences

Specifies to the *read_lib* command the maximum number of differences to list between the v3.1 format description of a library cell and its statetable description.

Data Types

integer

Default -1 in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Specifies to the *read_lib* command the maximum number of differences to list between the v3.1 format description of a library cell and its statetable description. The default value of -1 allows all differences to be listed.

For example, if *libgen_max_differences* = 5, *read_lib* lists only up to 5 differences between a library cell's v3.1 format description and its statetable description.

To see the value of this variable, type *printvar libgen_max_differences*. For a list of all *io* variables and their values, type *print_variable_group io*.

Feedback

See Also

- read_lib

---

libsetup_max_auto_opcond_message

Controls the number of inferred operating condition messages issued.

Data Types

int

Default 10

Group

mv

Description

This variable controls the total number of LIBSETUP-751 messages to be reported. By default, the integer variable is set to 10 and only first 10 LIBSETUP-751 messages are reported. The variable only has an effect if the operating conditions for pad/macro/switch cells have been inferred. The variable can be set to any integer number to control the message reporting.

Examples

```
set libsetup_max_auto_opcond_message 14             # reports 14
 LIBSETUP-751 messages

set libsetup_max_auto_opcond_message 0              # doesn't report
 LIBSETUP-751

set libsetup_max_auto_opcond_message 100            # reports 100
 LIBSETUP-751 messages
```

See Also

- LIBSETUP-751
- LIBSETUP-754

---

link_allow_design_mismatch

Allows the tool to modify the netlist to circumvent design mismatches during the link process.

This variable is supported only by DC Explorer.

Feedback

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

true in DC Explorer

Description

This variable allows DC Explorer to modify the netlist to circumvent design mismatches and to successful link the design during the link process. Formal verification is not validated.

This variable is supported only by DC Explorer. It is ignored by the Design Compiler tool and Design Compiler topographical mode.

To determine the current setting of this variable, use the *printvar link_allow_design_mismatch* command. For a list of all system variables and their current settings, use the *print_variable_group* system command.

See Also

- report_design_mismatch

---

link_allow_physical_variant_cells

Allows the Design Compiler tool to map physical variant cells to their master cell in the library during linking.

Data Types

Boolean

Default false

Description

When the *link_allow_physical_variant_cells* variable is set to true, the Design Compiler tool maps physical variant cells in a netlist or a DEF file to the master cell in the library. During linking, identifies the master library cells of these physical variant cells from the library and maps the physical variant cells to their master library cell.

Feedback

link_allow_pin_name_synonym

Allows the tool to link a cell containing pin names that do not match the pin names of the target library cell during the link process.

This variable is supported only in DC Explorer.

Data Types

Boolean

Default true

Description

Setting this variable to true allows the tool to link a cell when a pin name does not match the pin name of the target library cell. If the design has an instance of a subdesign with ports a, b, and z, but the link library provides the matching cell that has a target cell with ports a, b, and sum, the link process fails by default. However, when you set this variable to true and use the *set_pin_name_synonym z sum* command to indicate that the z and sum ports are equivalent, the link process succeeds. Note that link_allow_design_mismatch must also be set to true to enable this feature.

This variable is supported only in DC Explorer. It is ignored by Design Compiler and Design Compiler topographical mode.

To determine the current setting of this variable, use the *printvar link_allow_pin_name_synonym* command. For a list of all system variables and their current values, use *print_variable_group system*.

See Also

- set_pin_name_synonym

link_allow_upf_design_mismatch

Allows the usage of design data with inconsistencies or errors so that an approximate analysis can be completed in the tool.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

true in DC Explorer

Feedback

Group

mv

Description

This variable, when set to *true* (the default), allows the tool to accept design data containing inconsistencies or errors that would otherwise prevent completion of tasks in the tool.

A *true* setting allows inconsistent data in the following ways:

- *mv_no_main_power_violations* variable: behavior as if set to false (irrespective of actual setting)
- *mv_use_std_cell_for_isolation* variable: behavior as if set to true (irrespective of actual setting)
- *link* command allows selection of cells with PVT parameters that fail to meet requirements

When this variable is set to *false*, each individual feature is controlled by the setting of the feature-specific variable, and the *link* command requires PVT parameters to be correct.

See Also

- mv_no_main_power_violations
- mv_use_std_cell_for_isolation

---

link_force_case

Controls the case-sensitive or case-insensitive behavior of the *link* command.

Data Types

string

Default check_reference in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

case_insensitive in DC Explorer

Description

Controls the case-sensitive or case-insensitive matching policy to be used on HDL identifiers considered during a *link* command. The value of this variable can be set to *check_reference* (the default), *case_sensitive*, or *case_insensitive*.

Feedback

By default, the reference and the design cells to be linked are checked to ascertain the alphabet-case matching policy of the HDL input format that created that particular cell. Then, the least-sensitive matching rule is applied to determine whether the design cell resolves the reference. For example, a VHDL reference is linked case-insensitively and a Verilog reference is linked case-sensitively, while mixed language linkage candidates match case-insensitively. To override this behavior, you can set the value of the *link_force_case* variable to either *case_sensitive* or *case_insensitive*.

The appropriate matching policy applies to all the HDL identifiers that must be matched by the link: design template, port, and parameter names. It does not apply to the (canonical forms of) data values in any parameter override, which (for string data) are always case-sensitive. It also does not apply to library and architecture identifiers, which are always case-insensitive. When link candidates are built by elaboration, case-forcing does not change the HDL identifiers of the design or reference cells that are built.

Setting this variable to either the *case_sensitive* or the *case_insensitive* value can lead to results which are inconsistent with the default rules, since a forced rule applies throughout the linked hierarchy.

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

See Also

- link

---

link_library

Specifies the list of design files and libraries used during linking.

Data Types

list

Default * your_library.db

Description

This variable specifies the list of design files and libraries used during linking. The *link* command looks at the files and tries to resolve references in the order of the specified files. A "*" entry in the value of this variable indicates that the *link* command is to search all the designs loaded in dc_shell while trying to resolve references. If file names do not include directory names, files are searched for in the directories in *search_path*. The default is {"*" your_library.db}. Change *your_library.db* to reflect your library name.

To determine the current value of this variable, use the *printvar link_library* command. For a list of all system variables and their current values, use the *print_variable_group system* command.

Feedback

See Also

- link

---

link_portname_allow_period_to_match_underscore

Enables the linker to allow a period (.) as an alternative to an underscore (_) when doing port name matching.

Data Types

Boolean

Default false

Description

During linking, if named port mapping is used in the cell instance statement, the linker resolves port connections based on the port names. If the linker is unable to find a matching port name, and the *link_portname_allow_period_to_match_underscore* variable is set to *true*, the linker replaces, for comparison purposes, the "." characters in the port name from the cell instance statement with the "_" character to see if there is a match. This variable has no effect when positional port mapping is used instead.

The following Verilog example shows the effect of this variable on the *link* command:

```
module top ( B , A, Y);
      input A,B;
      output Y;
      wire Y;
      mid mid1 (.\\B.X (B), .A(A), .Y(Y));
    endmodule

    module mid (B_X, A, Y);
      input B_X;
      input A;
      output Y;
      assign Y = B_X & A;
    endmodule
```

The following scenarios assume that you have already read in the Verilog file and are running the *link* command.

Scenario 1:

link_portname_allow_period_to_match_underscore set to *false*

The linker attempts to resolve the port connections but is unable to find a port with the name "B.X" in the port list of the mid design. This causes the linker to issue a LINK-1 error.

Feedback

```
Error: Can't find inout port 'B.X' on reference to 'mid' in 'top'.
 (LINK-1)
```

This is expected because the instantiation of mid1 uses B.X as the port name but the actual port name is B_X.

Scenario 2:

link_portname_allow_period_to_match_underscore set to *true*

The linker first attempts to resolve the port connections but fails to resolve the "B.X" port name as in the previous scenario. However, since the *link_portname_allow_period_to_match_underscore* variable is set to *true*, the linker takes "B.X" and replaces all '.' characters with the '_' character. For this comparison, the linker will compare "B_X" against "B_X". In this case, the port names match and the design links successfully.

See Also

- link

---

link_portname_allow_square_bracket_to_match_underscore

Enables the linker to allow a square bracket([]) as an alternative to an underscore (_) when doing port name matching.

Data Types

Boolean

Default false

Description

During linking, the linker resolves port connections based on port names when named port mapping is used in cell instantiation. In the default mode, the linker looks for an exact match of the port names. Setting the *link_portname_allow_square_bracket_to_match_underscore* variable to *true*, allows the linker to also match the "[]" characters with the "_" character. This variable has no effect when positional port mapping is used. This variable should be used only for matching modport arrays.

The following SystemVerilog example shows the effect of this variable on the *link* command. *Note* - this example requires another related variable *link_portname_allow_period_to_match_underscore* to also be set to true. This is because this example contains both "[]" and "." characters in the original portname.

```
interface AA;
```

Feedback

```
    wire valid;
    wire ready;

    modport slave (input valid, output ready);
    modport master (output valid, input ready);

    endinterface

    module subblk (
     AA.master aa_master[0:3],
     AA.slave aa_slave[0:3]
    );

    // subblk logic
    assign aa_master[0].valid=aa_slave[0].valid & aa_slave[1].valid ;
    assign aa_master[1].valid=aa_slave[1].valid & aa_slave[2].valid ;
    assign aa_master[2].valid=aa_slave[2].valid & aa_slave[3].valid ;
    assign aa_master[3].valid=aa_slave[3].valid & aa_slave[0].valid ;
    endmodule


    module top ( input clk_a, input reasrt_n);
      AA aa_vec[0:3] ();
      subblk U_subblk ( .aa_master
(aa_vec.master), .aa_slave(aa_vec.slave));
    endmodule
```

The following scenarios assume that you have already read in the SystemVerilog file and are running the *link* command.

Scenario 1:

link_portname_allow_square_bracket_to_match_underscore set to **false**
link_portname_allow_period_to_match_underscore set to **false**

The linker attempts to resolve the port connections but is unable to find a port with the name "aa_master[0].valid" in the port list of the subblk design. This causes the linker to issue a LINK-1 error.

```
Error: Can't find inout port 'aa_master[0].valid' on reference to
 'subblk' in 'top'. (LINK-1)
```

This is expected because the instantiation of 'subblk' uses 'aa_master[0].valid' as the port name but the actual port name is 'aa_master_0__valid'.

Scenario 2:

link_portname_allow_square_bracket_to_match_underscore set to **true**
link_portname_allow_period_to_match_underscore set to **true**

The linker now matches all '.', and '[]' characters with the '_' character, and "aa_master[0].valid" matches "aa_master_0__valid", and the design links successfully.

Feedback

See Also

- link_portname_allow_period_to_match_underscore
- link

---

link_preserve_dangling_pins

Connects dangling pins to dummy cells during optimization to prevent the pin logic from being removed.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

false in DC Explorer

Description

Setting this variable to true allows the tool to connect dangling pins to dummy cells during optimization so that the pin logic does not get removed. This prevents constant propagation because of the removal of the pin logic.

See Also

- link_allow_design_mismatch

---

logic_level_report_group_format

Controls the format of the columns to be displayed during the Logic level path report section of the *report_logic_levels* command.

Data Types

string

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

%slack %ll %llthreshold %startpoint %endpoint in DC Explorer

Feedback

Description

This variable controls the format of the columns to be displayed during the *report_logic_levels* command.

The default specification is shown in the DEFAULT section above. The headings and order of the columns displayed correspond to the keywords specified in the syntax. For example, "%slack" specifies the WNS column, "%ll" the LOGIC LEVELS column, and so on.

Table 1

Default Output Format

```
            LOGIC
     WNS    LEVELS  THRESHOLD START POINT                  END POINT
  --------  ------  --------  --------------------------
 -------------------------
   -2.3006   13        5       REG_BLK/DATA_OUT_reg[0]/CK
 UPC_BLK/DATA_OUT_reg[7]/D
   -2.2972   11        5       REG_BLK/DATA_OUT_reg[0]/CK
 UPC_BLK/DATA_OUT_reg[3]/D
   -2.2899   10        5       REG_BLK/DATA_OUT_reg[0]/CK
 UPC_BLK/DATA_OUT_reg[3]/D
   -2.2891   11        5       REG_BLK/DATA_OUT_reg[0]/CK
 UPC_BLK/DATA_OUT_reg[9]/D
   -2.2879    2        5       REG_BLK/DATA_OUT_reg[0]/CK
 UPC_BLK/DATA_OUT_reg[5]/D
```

There are 10 possible columns that can be displayed; only 5 are displayed in the default format. You can create a customized output format by specifying any number of the available columns.

Following are the available columns and their header values.

```
%slack              WNS
%ll_buf_inv         ALL LOGIC LEVELS
%ll                 LOGIC LEVELS
%llthreshold        THRESHOLD
%startpoint         START POINT
%endpoint           END POINT
%clockcycles        CLOCK CYCLES
%startclk           START CLOCK
%endclk             END CLOCK
%infeas             INFEASIBLE
```

Examples

The following example sets the group format to report all the columns

```
prompt> set logic_level_report_group_format "%slack %ll_buf_inv %ll \
%llthreshold %startpoint %endpoint %clockcycles %startclk %endclk
 %infeas"
```

Feedback

See Also

- set_analyze_rtl_logic_level_threshold
- report_logic_levels
- logic_level_report_summary_format

---

logic_level_report_summary_format

Controls the format of the columns to be displayed during the Summary section of the *report_logic_levels* command.

Data Types

string

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

%group %period %wns %num_paths %max_level in DC Explorer

Description

This variable controls the format of the columns to be displayed during the *report_logic_levels* command.

The default specification is shown in the DEFAULT section above. The headings and order of the columns displayed correspond to the keywords specified in the syntax. For example, "%group" specifies the GROUP column, "%period" the REQUIRED PERIOD column, and so on.

Table 1

Default Output Format

GROUP	REQUIRED PERIOD	WNS	NUM OF PATHS	MAX LEVEL
All Path Groups	n/a	n/a	2310	13
CLOCK	0.0100	-2.3289	1600	13
custom_group1	0.0100	-2.3174	166	12
custom_group2	0.0100	-2.3006	544	13

There are 8 possible columns that can be displayed; only 5 are displayed in the default format. You can create a customized output format by specifying any number of the available columns.

Following are the available columns and their header values.

Feedback

```
%wns            WNS
%group          GROUP
%period         REQUIRED PERIOD
%std_dev        STANDARD DEVIATION
%num_paths      NUM OF PATHS
%max_level      MAX LEVEL
%min_level      MIN LEVEL
%avg_level      AVERAGE LEVEL
```

Examples

The following example sets the summary format to report all the columns

```
prompt> set logic_level_report_summary_format "%wns %group %period
 %std_dev \
%num_paths %max_level %min_level %avg_level"
```

See Also

- set_analyze_rtl_logic_level_threshold
- report_logic_levels
- logic_level_report_group_format

---

ltl_obstruction_type

Controls the routing blockage type for the named obstructions, without the route type being specified.

Data Types

Boolean

Default placement_only in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable controls the routing blockage type for the named obstructions, without the route type being specified. When the setting is *placement_only* (the default), the obstructions are treated as placement obstructions only and routing wires can still go through. When the setting is *routing_none*, the obstructions are treated as routing blockages and no routing wires are allowed.

To determine the current value of this variable, use the *printvar ltl_obstruction_type* command. For a list of all *links_to_layout* variables and their current values, use *print_variable_group links_to_layout*.

See Also

- lbo_cells_in_regions

m

magnet_placement_disable_overlap

Controls whether the *magnet_placement* command can move pre-placed cells during magnet placement.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode and Design Compiler NXT topographical mode

Description

By default, the *magnet_placement* command moves the available pre-placed cells in the chip area during magnet placement.

If the *magnet_placement_disable_overlap* variable is set to true, the location of pre-placed cells is kept. The *magnet_placement* command looks for other legal locations to place magnet objects. If no legal location is found, the magnet objects are not pulled closer to the specified magnet.

See Also

- get_magnet_cells
- magnet_placement

magnet_placement_fanout_limit

Sets the threshold of the high-fanout nets during the *magnet_placement* command.

Feedback

Data Types

integer

Default Variable is not supported in Design Compiler non-topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

1000 in Design Compiler topographical mode and Design Compiler NXT topographical mode

Description

The *magnet_placement_fanout_limit* variable controls whether the nets with high fanout are pulled during magnet placement. If the net has more fanouts than the specified threshold value, the cells on the net are not pulled.

See Also

- get_magnet_cells
- magnet_placement

---

magnet_placement_stop_after_seq_cell

Controls whether sequential cells are pulled towards a specified magnet during magnet placement.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode and Design Compiler NXT topographical mode

Description

This variable controls whether sequential cells are pulled towards a specified magnet during magnet placement.

When *false* (the default), sequential cells are not pulled towards a specified magnet when you run the *magnet_placement* command with the *-stop_by_sequential_cells* option.

When *true*, sequential cells are pulled towards a specified magnet when you run the *magnet_placement* command with the *-stop_by_sequential_cells* option.

By default, the magnet placement operation terminates before sequential cells when you run the *magnet_placement* command with the *-stop_by_sequential_cells* option.

Feedback

See Also

- get_magnet_cells
- magnet_placement

---

mcmm_high_capacity_effort_level

Controls the behavior of multicorner-multimode (MCMM) scenario reduction.

Data Types

Float. Valid range is between 0 and 10.

Default 0 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Group

MCMM

Description

This variable controls how aggressively the multicorner-multimode scenario reduction feature reduces the number of dominant scenarios.

By default, any scenario with a violation that is worst across all scenarios is included in the dominant set of scenarios. As you increase the value of this variable, scenario reduction adjusts its criteria for comparing the slack of a violating object across different scenarios. This allows for further reduction of the dominant scenario set, but implies that a few violations will not be fixed during optimization.

If you set this variable to a value smaller than 0, scenario reduction is performed using an effort level of 0. If you set this variable to a value larger than 10, scenario reduction is performed using an effort level of 10.

See the man page for the *get_dominant_scenarios* command for more information about scenario reduction.

See Also

- all_active_scenarios
- all_scenarios
- set_active_scenarios

Feedback

mpc_area_for_max_sized_cells

When set, considers the largest size lib cells to compute area during auto-floorplan generation.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

true in Design Compiler NXT topographical mode

Description

When variable *mpc_area_for_max_sized_cells* is set to *true* before *compile_ultra -spg*, and auto-floorplan is triggered, the tool computes the core area based on the largest size libcell present in the given libraries. This might help small designs to accommodate any kind of sizing and buffering. This way, tool ensures that the cells are accommodated inside the die even after sizing to biggest lib cell.

The variable takes effect for small designs that have cell count less than or equal to 50.

This feature in supported in Design Compiler NXT topographical mode.

Examples

The following example shows how to enable and disable the variable:

```
prompt> set mpc_area_for_max_sized_cells true
prompt> set mpc_area_for_max_sized_cells false
```

See Also

- dcnxt_use_icc2_link
- set_auto_floorplan_constraints
- report_auto_floorplan_constraints

mux_auto_inferring_effort

Specifies the MUX inferring effort level.

Data Types

integer

Feedback

Default 2 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

The *mux_auto_inferring_effort* variable controls the MUX inferring effort of Design Compiler FPGA. Valid values are *0* through *6*. The default value is *2*. The larger the integer value, the more MUX is inferred.

---

mv_align_library_pg_pins

Set this variable to get a consistent ordering of power pins across multicorner-multimode libraries.

Data Types

Boolean

Default true

Group

none

Description

When this variable is set to true, all libraries across multicorner-multimode will have a consistent ordering of power pin.

---

mv_allow_buf_on_mv_boundary_nets

Allow buffering on the nets between mv cells and the power domain boundaries associated with the strategies of those cells

Data Types

Boolean

Default false

Description

By default, the tool sets a persistent dont_touch marking on nets between mv cells (isolation cells, level shifters and enable level shifters) and the power domain boundaries associated with their strategies.

Feedback

With this variable set to *true*, this restriction will be relaxed during optimization commands which insert buffers (compile_ultra, optimize_netlist) in order to permit buffering the nets between the mv cells and their associated power domain boundaries. With these buffers in the nets between the MV cells and their associated power domain boundaries, the MV cells association should remain intact.

Please note: When user sets this appOption to be true, user should still expect to see dont_touch derived on the nets between the MV cells and power domain boundaries associated with the strategies. The dont_touch restriction is relaxed and re-propogated before and after the buffering step to allow buffering to happen.

---

mv_allow_force_ls_with_iso_violations

Allows the insertion of level-shifters specified by means of a *set_level_shifter -force_shift* strategy, despite an Isolation violation.

Data Types

Boolean

Default true

Description

This variable controls the insertion of level-shifters specified by *set_level_shifter -force_shift* strategies on paths with Isolation violation. This behavior is true by default.

To prevent the tool from inserting *-force_shift* level-shifters on nets with isolation violation, set this variable to *false*.

See Also

- set_level_shifter
- insert_mv_cells

---

mv_allow_ls_on_leaf_pin_boundary

Allows level-shifter insertion on leaf pin (such as macro cell pin) boundaries.

Data Types

Boolean

Default false

Group

mv

Feedback

Description

This variable controls whether or not to allow level-shifter insertion on leaf pin boundaries that are not power domain boundaries. For example, When a macro cell is operating at a voltage different from its surrounding logic, and you want level shifters to be inserted at the interface, the recommended flow is to define a power domain around the macro cell by specifying the macro cell as the root cell of a power domain.

If you do not define the macro cell as the root cell, level shifters are not inserted at the interface, because the interface is not a power domain boundary. By default, level shifters are only inserted at power domain boundaries.

If you are unable to define a power domain around the macro cell, but still require level shifters to be inserted, or if you need level shifters at the interface of any kind of leaf cell, use this variable to enable the non-default behavior.

See Also

- compile_ultra

---

mv_allow_ls_per_macro_fanout

Set this variable to insert level shifter on each macro fanout load pin on a level shifter violating path.

Data Types

Boolean

Default false

Group

none

Description

When this variable is set to true, each macro fanout load pin will get level shifter, even when macro pins have same related supply. DC always try to insert minimum number of level shifters on a violating path. So a path ending with multiple marco laods will get single level shifter if all have same related supply. But with this variable set to true each load will get one level shifter.

---

mv_allow_ls_per_output_port

Set this variable to insert level shifter on each output ports on a level shifter violating path.

Feedback

Data Types

Boolean

Default false

Group

none

Description

When this variable is set to true, each output port load pin will get level shifter, even when output ports have same related supply. DC always try to insert minimum number of level shifters on a violating path. So a path ending with multiple output ports will get single level shifter if all have same related supply. But with this variable set to true each load will get one level shifter.

---

mv_allow_multiple_power_domain_in_voltage_area

Allow multiple power domain puts into single voltage area.

Data Types

Boolean

Default false

Group

MV-UPF

Description

1. When the variable value is false, ICC requests power domain and voltage area 1-to-1 mapping.\n (a) The top most power domain is always associated with DEFAULT_VA. \n (b) Other power domain and voltage area are 1-to-1 mapping with the same name and elements.\n \n Example 1:\n create_power_domain PD_TOP -include_scope\n create_power_domain PD_SUB -elements {sub}\n create_power_domain PD_BLK -elements {blk} -scope {blk}\n \n create_voltage_area -power_domain {PD_SUB}\n create_voltage_area -power_domain {blk/PD_BLK}\n \n ICC usual UPF flow requests UPF power domain creation before voltage area creation. The above example is for ICC usual UPF flow. The design defines 3 power domains and 3 voltage areas. ICC derives a special voltage area DEFAULT_VA and auto associates it with power domain PD_TOP.\n \n Example 2:\n create_voltage_area -name {PD_SUB} {sub}\n create_voltage_area -name {blk/PD_BLK} {blk}\n \n create_power_domain PD_TOP -include_scope\n create_power_domain PD_SUB -elements {sub}\n create_power_domain PD_BLK -elements {blk} -scope {blk}\n \n The above example is recommended only for ICC UPF ODL flow, since ODL flow requests voltage area

Feedback

creation before UPF power domain creation. Please do not use that for ICC usual UPF flow. In ODL flow, users must make sure power domain and voltage area has the same name and elements. A special voltage area DEFAULT_VA will be derived and auto associated with power domain PD_TOP.\n \n 2. When the value is true, ICC allows multiple power domain associated with single voltage area.\n (a) The voltage area name must be different from any power domain of the whole design. \n (b) The multiple power domains must have the equivalent primary power and ground nets.\n (c) If the power domain is not defined in any voltage area, it will be associated with DEFAULT_VA.\n (d) The top most power domain must be associated with DEFAULT_VA.\n \n Example 3:\n create_power_domain PD_TOP -include_scope \n create_power_domain PD_SUB1 -elements {sub1}\n create_power_domain PD_SUB2 -elements {sub2}\n create_power_domain PD_BLK1 -elements {blk1} -scope {blk1}\n create_power_domain PD_BLK2 -elements {blk2} -scope {blk2}\n \n create_voltage_area -name {VA_SUB1_BLK2} -power_domain {PD_SUB1 blk2/PD_BLK2}\n create_voltage_area -power_domain {PD_SUB2}\n \n For the above example, the design has 5 power domains and 3 voltage areas. The power domain PD_SUB1 and blk2/PD_BLK2 are associated with voltage area VA_SUB1_BLK2. The power domain PD_SUB2 is associated with power domain PD_SUB2. The rest of the power domain PD_TOP and blk1/PD_BLK1 are associated with voltage area DEFAULT_VA.\n

```
prompt> echo $mv_allow_multiple_power_domain_in_voltage_area
```

See Also

- create_voltage_area
- report_voltage_area

---

mv_allow_pg_pin_reconnection

Allows command connect_supply_net to reconnect pg pins.

Data Types

Boolean

Default false

Group

mv

Description

This variable controls whether or not to allow reconnection on PG pins. By default, the connect_supply_net command errors out when connecting a supply net to a PG pin, which already has a supply net connection.

Feedback

By setting this variable to true, the connect_supply_net command reconnects the new supply net to the PG pin and the previous connection is discarded.

See Also

- connect_supply_net

---

mv_allow_upf_cells_without_upf

Enable the processing of pre-instantiated isolation cells without UPF context.

Data Types

Boolean

Default false

Group

mv

Description

When this variable is set to true, it allows compilation of designs which have pre-instantiated PM cells.

Currently, this kind of flow is supported only if there isn't any UPF command nor context, and only for pre-instantiated isolation cells.

mv_allow_upf_cells_without_upf cannot be set to true if there are power domains defined in any design, and consequently no power domain can be created while this variable is set to true.

---

mv_allow_va_beyond_core_area

Allows a voltage areas to be created outside of the core area.

Data Types

Boolean

Default false

Group

mv

Feedback

Description

This variable, when set to *true*, allows voltage areas to be created even when the specified voltage area is not fully enclosed inside core area. The default is *false*, which requires any new voltage areas to be created inside the core area.

See Also

- create_voltage_area

---

mv_disable_voltage_area_aware_detour_routing

This variable determines whether virtual routes used in analysis and optimization need to consider voltage areas as blockages. By default, a net from one voltage area (or the default voltage area) needs to detour around other voltage areas. Set this variable to *true* to ignore the presence of voltage areas.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

---

mv_input_enforce_simple_names

Enforces the use of simple names for restricted commands as per the IEEE 1801 (UPF) standard.

Data Types

Boolean

Default true

Group

mv

Description

This variable controls the use of hierarchical names for the arguments of certain restricted commands that require simple names as per the IEEE 1801 (UPF) standard.

The default value of this variable is true. So, by default, the tool accepts only simple names and errors out when you use hierarchical names for the arguments of certain

Feedback

restricted commands. This variable is honored when you specify the UPF input in ASCII and .ddc file formats.

See Also

- mv_output_enforce_simple_names

---

mv_insert_level_shifters_on_ideal_nets

Directs automatic level-shifter insertion to insert level shifters on ideal nets.

Data Types

string

Default ""

Group

none

Description

When this variable is set to "all", automatic level-shifter insertion inserts level shifters on all ideal nets that need level shifters.

By default, automatic level-shifter insertion does not insert level shifters on ideal nets.

Note that if a net is also a clock net, automatic level-shifter insertion does not insert any level shifter on it, unless the variable *auto_insert_level_shifters_on_clocks* is set to "all" or contains the net name.

See Also

- auto_insert_level_shifters
- auto_insert_level_shifters_on_clocks
- create_clock
- set_ideal_network

---

mv_make_primary_supply_available_for_always_on

Allows the primary supply of a power domain to be used for buffering always-on feedthrough nets passing through the domain.

Feedback

Data Types

Boolean

Default true

Group

mv

Description

This variable determines whether always-on synthesis can use the primary supplies of a power domain for buffering feedthrough nets that pass through the domain.

By default, the variable is set to *true*, which allows the primary supplies of a power domain to be used for buffering always-on feedthrough nets, as long as doing so does not introduce electrical violations. Buffers ordinarily used in the power domain are used as much as possible to get the best possible QoR.

If the variable is set to *false*, feedthrough nets are marked as always-on nets with the related supply from driver or loads, requiring that any buffers used on the feedthrough net be compatible with the driver or load domains. This option is provided mainly for backward compatibility with previous releases of the tool, which always had this requirement.

See Also

- get_always_on_logic
- analyze_mv_design

---

mv_mtcmos_detour_obstruction

Allows the connection of power switch cells to bypass obstructions.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Group

mv

Feedback

Description

This variable controls whether or not to detour obstructions such as blockages and macro cells when connecting power switch cells.

This variable setting is applicable only when you use the *connect_power_switch* command with the *-mode fishbone* option.

---

mv_no_always_on_buffer_for_redundant_isolation

Allows normal buffers to be used at nets driving the data input of redundant isolation cells.

Data Types

Boolean

Default false

Group

mv

Description

This variable controls whether or not always-on buffers are used on the nets driving the data-input pins of redundant isolation cells. If the variable is set to *true*, always-on buffer or inverter cells are avoided on these nets in favor of regular buffer or inverter cells supplied by the local power of the net's power domain. By default, any necessary buffering is done using the global driver supply of the net.

See Also

- set_isolation
- set_isolation_control

---

mv_no_cells_at_default_va

Controls whether the tool can place new buffers at the default voltage area.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Feedback

Description

This variable, when set to *false* (the default), allows the tool to place new buffers at the default voltage area.

When this variable is set to *true*, the buffers created during optimization are not allowed to be placed at the default voltage area.

Set this variable to *true* for a physically abutted design where there is no space in the default voltage area, or in other situations to prevent the tool from inserting new buffers at the default voltage area.

This variable is honored by the *place_opt*, *clock_opt*, and *route_opt* commands.

---

mv_no_main_power_violations

Selects Standard Cell Main Rail (SCMR) as the main power supply for level shifters.

Data Types

Boolean

Default true in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

false in DC Explorer

Group

mv

Description

This variable controls whether or not level shifters use SCMR as their main power supply. By default, level shifters must get their main power supply from SCMR.

To allow the tool to select other power sources as the main power supply for level shifters, set this variable to *false*.

See Also

- check_mv_design
- compile
- insert_mv_cells

Feedback

mv_output_enforce_simple_names

Enforces the use of simple names for restricted commands as per the IEEE 1801 (UPF) standard.

Data Types

Boolean

Default true

Group

mv

Description

This variable controls the printing of hierarchical names for the arguments of certain restricted commands in the UPF file or on the standard output device. The IEEE 1801 standard requires that the arguments of these commands accept only simple names.

The default value of this variable is true. So, by default, these restricted commands print only simple names for the arguments in the UPF file or on the standard output. The tool inserts the *set_scope* command appropriately to set the current scope at the proper hierarchy. This ensures that the use of a simple name is sufficient. After printing, the original scope is returned.

See Also

- mv_input_enforce_simple_names

mv_output_upf_line_indent

Sets the number of indentation spaces inserted at the beginning of each line when *save_upf* splits long commands onto multiple lines.

Data Types

integer

Default 2

Description

When line splitting for the *save_upf* command is enabled, this variable specifies the number of spaces added to the beginning of each line written, except for the first line of each command (which is not indented).

Feedback

Line splitting for the *save_upf* command is controlled by the *mv_output_upf_line_width* variable.

When line splitting for the *save_upf* command is disabled, the *mv_output_upf_line_indent* variable has no effect.

See Also

- save_upf
- mv_output_upf_line_width

---

mv_output_upf_line_width

Specifies whether the *save_upf* command writes out long commands as multiple lines, and if so, the threshold for splitting lines.

Data Types

Integer

Default 0

Description

When this variable is set to 0, line splitting for *save_upf* is turned off. The command writes out each UPF command as a single line, regardless of its length.

If this variable is set to a positive value, every command that is longer than that threshold is written out in multiple lines, each line containing no more than the specified number of characters, and linked by the backslash character at the end of each continuing line. You cannot set this variable to a negative number.

The second and subsequent lines of a UPF command are indented (offset to the right) by the number of spaces determined by the *mv_output_upf_line_indent* variable.

See Also

- save_upf
- mv_output_upf_line_indent

---

mv_skip_opcond_checking_for_unloaded_level_shifter

Skips operating condition checking for level-shifter cells with unconnected output pins.

Data Types

Boolean

Feedback

Default false

Group

mv

Description

This variable controls the checking of operating condition on level-shifter cells with unconnected output pins.

By default, operating condition checking is performed on all level-shifter cells in the design.

For the tool to skip operating condition checking for level-shifter cells with unconnected output pins, set this variable to *true*.

See Also

- check_mv_design
- compile
- insert_mv_cells
- set_opcond_inference

---

mv_upf_enable_forward_bias_check

Controls whether or not implementation of designs using well bias will allow forward well bias

Data Types

Boolean

Default false

Group

mv

Description

When the variable is set to true the tool will test for voltage differences between power/ground supplies and its related well bias supply to detect forward well bias occurrences. In addition, the tool will try to avoid such occurrences by making use of well bias insulated cells during cell insertion and mapping.

Feedback

See Also

- check_mv_design
- mv_upf_enable_forward_reverse_bias_check
- mv_upf_enable_reverse_bias_check

---

mv_upf_enable_forward_reverse_bias_check

Controls whether or not implementation of designs using well bias will allow forward and reverse well bias

Data Types

Boolean

Default false

Group

mv

Description

When this variable is set to true the tool will test for voltage differences between power/ground supplies and its related well bias supply to detect forward or reverse well bias occurrences. In addition, the tool will try to avoid such occurrences by making use of well bias insulated cells during cell insertion and mapping.

See Also

- check_mv_design
- mv_upf_enable_forward_bias_check
- mv_upf_enable_reverse_bias_check

---

mv_upf_enable_reverse_bias_check

Controls whether or not implementation of designs using well bias will allow reverse well bias

Data Types

Boolean

Default false

Feedback

Group

mv

Description

When the variable is set to true the tool will test for voltage differences between power/ground supplies and its related well bias supply to detect reverse well bias occurrences. In addition, the tool will try to avoid such occurrences by making use of well bias insulated cells during cell insertion and mapping.

See Also

- check_mv_design
- mv_upf_enable_forward_reverse_bias_check
- mv_upf_enable_forward_bias_check

---

mv_upf_tracking

Controls whether the UPF tracking feature is enabled in the current session.

Data Types

Boolean

Default true

Description

This variable controls whether the UPF tracking feature is enabled in the current session. The default value of this variable is *true*.

When this variable is set to *true* (the default), the user-specified UPF commands are tracked to ensure that their original order is preserved. Also, the UPF' file written by the tool contains two sections. The first section contains the user-specified commands and the second section contains the tool-generated commands. The beginning of the tool-generated section is marked by the following variable setting:

```
set derived_upf true
```

Similarly the end of the tool-generated section is marked by the following variable setting:

```
set derived_upf false
```

To ensure that user-specified UPF content is preserved, this variable has the following usage restrictions:

1. Changing the value of this variable from *true* to *false* is allowed. However, when the variable is changed from *true* to *false*, the variable is marked read-only. This is to ensure that the variable cannot be changed to *true* again.
2. Changing the value of this variable from *false* to *true* is not allowed when the design has UPF constraints.

---

mv_use_std_cell_for_isolation

Allows the tool to insert standard cells as UPF isolation cells.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

true in DC Explorer

Group

mv

Description

This variable determines whether standard cells can be used as isolation cells. When the variable is set to *false* (the default), standard cells are not used as isolation cells.

When the variable is set to *true*, a latch or a two-input NAND, AND, OR, or NOR gate can be used as an isolation cell if the *ok_for_isolation* lib_cell attribute is set to *true* for the cell, and the *isolation_cell_enable_pin* lib_pin attribute is set to *true* on the enable pin. Based on the isolation sense and clamp value of the isolation strategy, the tool selects an appropriate standard cell for use as an isolation cell and connects it to the isolation supply.

See Also

- set_isolation
- set_isolation_control

---

mw_cell_name

Contains the Milkyway design cell name.

Feedback

Data Types

string

Default ""

Description

This variable contains the Milkyway design cell name. By default, the string is empty. If given, *read_mdb* reads in the cell specified by this variable, or *read_mdb* modifies this variable to indicate the cell, view, and version of the MDB cell read in. The *write_mdb* command searches for this variable when writing data to an MDB cell. If this variable is set, it writes to the cell specified by this variable, or it writes to a cell using the current design name.

---

mw_design_library

Contains the Milkyway design library.

Data Types

string

Default ""

Description

This variable contains the Milkyway design libraries. The default is the empty string. When this variable is set, the Milkyway design is stored in the directory specified by the variable. If this variable is set, it will be used by the *read_milkyway*, *write_milkyway* and *create_mw_design* commands.

This variable needs to be set for a smooth data transfer to Milkyway.

See Also

- write_milkyway

---

mw_disable_escape_char

Disables the escape characters for hierarchy delimiters.

Data Types

Boolean

Default true

Feedback

Description

This variable controls the behavior of escape characters for hierarchies. This variable is used only by the *read_mdb* and *write_mdb* commands.

By default, the tool always escapes the hierarchy delimiters if they are part of names.

When set to *false*, it enables escape characters for hierarchies, similar to setting "No Backslash Insertion to avoid Hier Name Collisions" in Astro.

Make sure that the design does not have name collisions.

---

mw_hdl_bus_dir_for_undef_cell

Specifies how to determine the bus direction for an undefined cell.

Data Types

integer

Default 0

Description

If any port of any undefined cell is a bus, verilog2cel Verilog reader creates a bus port for the undefined cell and instantiates the cell. This variable specifies the direction of the bus.

For example, assume an undefined cell is instantiated similar to the following:

```
sub si (.port(net[x:y]),...);
```

The number of bus bits = r = |x-y|+1 defines the bus origin according to the value given:

```
0 (From Connection)
x > y type [r:0] port
x < y type [0:r] port
```

1 (Descending)

type [r:0] port

2 (Ascending) type [0:r] port

To determine the current value of this variable, use the *printvar variable_name_filler* command. For a list of all HDL variables and their current values, use *print_variable_group hdl*.

See Also

- enable_cell_based_verilog_reader

Feedback

mw_hdl_expand_cell_with_no_instance

Determines whether or not to expand netlist cells without instances.

Data Types

Boolean

Default false

Description

This variable determines whether or not the verilog2cel Verilog reader expands netlist cells that do not contain physical descriptions. By default, no expansion occurs when you have netlist instances with no physical description. Set this variable to *true* if you have netlist instances containing only nets.

To determine the current value of this variable, use the *printvar mw_hdl_expand_cell_with_no_instance* command. For a list of all HDL variables and their current values, use *print_variable_group hdl*.

See Also

- enable_cell_based_verilog_reader

mw_reference_library

Contains the Milkyway reference libraries.

Data Types

list

Default ""

Description

This variable contains the Milkyway reference libraries. The default is the empty string. By setting this variable, the search_path and the physical_library will be enhanced to use the Milkyway libraries.

See Also

- create_mw_lib
- open_mw_lib

Feedback

mw_site_name_mapping

Specifies pairs of site names that synchronize a floorplan's site name with a physical library's site name.

Data Types

string

Default ""

Group

physopt

Description

This variable specifies pairs of site names that synchronize a floorplan's site name with a physical library's site name. When you use this variable, the floorplan site name is mapped to the physical library site name during library loading. You must set the variable before library loading.

The syntax for the variable is as follows:

```
set mw_site_name_mapping
{ {floorplan_site_name1 mw_reference_lib_site_name}
  {floorplan_site_name2 mw_reference_lib_site_name} }
```

You must use curly braces in the syntax in IC Compiler; therefore, you should also use curly braces in Design Compiler for compatibility between the tools.

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

Examples

The following example shows three site names in the floorplan file remapped to unit in the Milkyway reference library. You must specify the exact names in the correct case.

```
prompt> set mw_site_name_mapping { {CORE unit} {core unit} {core2 unit} }
```

n

ndm_load_mol_routing_layers

Enables loading the geometries for metal0 and via0 layers

Feedback

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

When variable *ndm_load_mol_routing_layers* is set to *true* before loading physical libraries, the tool loads the pin and obstruction geometries for metal0 and via0 layers.

This feature in supported in Design Compiler NXT topographical NDM mode.

Examples

The following example shows how to enable and disable loading of geometries at metal0/via0 layers:

```
prompt> set ndm_load_mol_routing_layers true
prompt> set ndm_load_mol_routing_layers false
```

See Also

- shell_is_in_ndm_mode

---

ndm_reference_library_from_mw

Specifies output directory where the NDM libraries converted from Milkyway libraries for advance nodes are saved.

Data Types

String

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

"" in Design Compiler NXT topographical mode

Description

Commands *create_mw_lib* and *create_lib* are enhanced to accept *Milkyway libraries*, and automatically triggers a flow to convert given *Milkyway libraries* to NDM libraries. These generated *IC Compiler II* NDM reference libraries are saved in a default directory *auto_ndms_from_mw* in the current working directory.

Feedback

The feature needs one of the ICC shell or Milkyway executable to read the given Milkyway libraries. Following tcl variables can be used to specify the ICC shell and Milkyway respectively before invoking commands *create_mw_lib* and *create_lib*: *dcnxt_ndm_library_configuration_icc_shell dcnxt_ndm_library_configuration_mw_exec*

The NDM generation flow is trigger via *create_mw_lib* command, only when tool understands that the provided libraries and technology file are for advance nodes that is 5nm and below. *create_lib* command has no technology node restriction.

Additionally, user can specify an output directory by setting variable *ndm_reference_library_from_mw* before invoking *create_mw_lib* or *create_lib* command. If specified, the NDM reference libraries converted from Milkyway libraries for advance nodes, will be saved in the specified directory.

This feature in supported in Design Compiler NXT topographical mode.

Examples

The following example shows how to set output NDM directory

```
prompt> set ndm_reference_library_from_mw "AutoNDMs"
```

See Also

- create_mw_lib
- create_lib
- shell_is_in_ndm_mode

---

net_auto_layer_promotion_max_utilization

Defines the maximum utilization of top metal layers that automatic layer promotions is allowed to use.

Data Types

float

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

1 in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

This variable controls the maximum utilization that the automatic layer promotion optimization is allowed to use for the top metal layers. Setting this variable to a lower value is likely to result in less nets that are promoted to top metal layers.

Feedback

The value of the variable is within range (0.0, 1.0], and the default is *1.0*, which corresponds to allowing 100% utilization of available resources for layer promotions.

To determine the current value of this variable, use the following command:

```
prompt> printvar net_auto_layer_promotion_max_utilization
```

See Also

- set_congestion_options

O

---

optimize_area_ignore_path_group_weights

Ignores the weights on path groups during area optimization in *optimize_netlist -area*.

Data Types

Boolean

Default true

Description

When *optimize_area_ignore_path_group_weights* variable is *true*, the *optimize_netlist -area* command ignores the weights set on the path groups. The weights of all path groups are set as 1 during area optimization. After area optimization is done, the weight on each path group is restored to the original value.

Examples

The following examples show how to enable and disable ignoring the weights on the path groups during area optimization for a design:

```
prompt> set optimize_area_ignore_path_group_weights true
prompt> set optimize_area_ignore_path_group_weights false
```

See Also

- optimize_netlist
- group_path

Feedback

optimize_ndr_critical_range

Sets the critical range value for choosing nets for nondefault routing rules during optimization.

Data Types

Float

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

0.008 in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Group

syn-qor

Description

Sets the critical range value for nondefault routing rules during optimization. The nets that fall within the following range are considered for nondefault routing rules:

[WNS, WNS*(1-optimize_ndr_critical_range)]

By default, the variable is set to 0.008. Therefore, if the worst negative slack is -1000, the range is [-1000, -992] where -992 = -1000*(1-0.008). If the delay value falls into this range, it is considered critical.

If you specify both the *optimize_ndr_critical_range* and *optimize_ndr_max_nets* variables, the tool uses the tighter constraints.

See Also

- compile_ultra

optimize_ndr_max_nets

Restricts the maximum number of nets to be assigned nondefault routing rules during optimization.

Data Types

Integer

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

2000 in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Group

syn-qor

Description

Sets a limit on the maximum number of nets to be assigned nondefault routing rules during optimization in the Design Compiler Graphical tool. Use the variable before running the *compile_ultra -spg* command.

See Also

- compile_ultra

---

optimize_ndr_user_rule_names

Enables the automatic timing-focused promotion of nets with nondefault routing rules in the Design Compiler Graphical tool.

Data Types

String

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

"" in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Set the *optimize_ndr_user_rule_names* variable to a prioritized list of nondefault routing rules that you want to use for automatic net promotion.

To define the nondefault routing rules use the *define_routing_rule command*. When you run the *compile_ultra* command, the tool automatically promotes nets based on the nondefault routing rules that you specified.

The default value for this variable is empty.

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

```
prompt> printvar optimize_ndr_user_rule_names
```

Feedback

See Also

- define_routing_rule
- compile_ultra

---

optimize_netlist_enable_state_reachability

Additional state-reachability to remove constant registers for the *optimize_netlist command.*

Data Types

Boolean

Default false

Description

When *optimize_netlist_enable_state_reachability* is *true*, the *optimize_netlist* command would do additional state-reachability to remove constant registers. The variable is useful if the constant propagates to the register only at the end of the flow.

See Also

- optimize_netlist

---

optimize_reg_add_path_groups

Instructs the *compile_ultra* command to create a group of paths on nets connected to retimed subdesigns.

Data Types

string

Default false

Description

This variable automatically creates path groups for nets inside of or connected to subdesigns that are retimed. Use this variable to create path groups for designs that are retimed using the *set_optimize_registers* command. This variable does not affect designs retimed using the *compile_ultra -retime* command.

When this variable is set to *inputs*, the *compile_ultra* command creates path groups for nets connected to input pins of the subdesign that is being retimed. A separate path group for each subdesign is created. Path groups are created at the beginning of compile and removed just before the retiming engine is called in the optimization flow.

Feedback

When this variable is set to *outputs*, the *compile_ultra* command creates path groups for nets connected to output pins of the subdesign that is being retimed. A separate path group for each subdesign is created. Path groups are created at the beginning of compile and removed just before the retiming engine is called in the optimization flow.

When this variable is set to *registers*, the *compile_ultra* command creates path groups for nets connected to data pins and next state pins of registers in the subdesign that is being retimed. A separate path group for each subdesign is created. Path groups are created at the beginning of compile and removed just before the retiming engine is called in the optimization flow. Use this setting to group paths that connect registers inside subdesigns being retimed.

By default, paths are not grouped. Set this variable to *false* to reset the variable to the default setting.

Be aware that paths grouped this way can result in longer runtime in Design Compiler.

To determine the current value of this variable, use the *printvar optimize_reg_add_path_groups* command. For a list of compile variables and their current values, use the *print_variable_group compile* command.

See Also

- compile_ultra
- set_optimize_registers
- group_path

---

optimize_via_ladder_critical_range

Sets the critical range value for choosing driver pins for via ladder insertion during optimization.

Data Types

Float

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

0.5 in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Group

syn-qor

Feedback

Description

Allows you to choose via ladders on driver pins in a timing path within a specific slack range during optimization. The variable sets the critical range value for via ladder insertion. The driver pins with nets falling within the following range are considered for via ladder insertion during optimization:

[WNS, WNS*(1-optimize_via_ladder_critical_range)]

By default, the variable is set to 0.5. Therefore, if the worst negative slack is -1000, the range is [-1000, -500] where -500 = -1000*(1-0.5). If the delay value falls into this range, it is considered critical.

See Also

- compile_ultra

---

optimize_via_ladder_net_length_threshold

Sets threshold on the length of nets connected to pins for via ladder assignment during optimization.

Data Types

integer

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

100 in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Group

syn-qor

Description

Sets a minimum length threshold on the nets considered for via ladder assignment during optimization in the Design Compiler Graphical tool. The value you specify is multiplied by the minimum height of the unit tile.

For example, if you specify a threshold value of 100, the minimum length threshold for nets considered for via ladder assignment is 100 times the minimum height of the unit tile.

See Also

- compile_ultra

Feedback

p

pdefout_diff_original

Writes cells not in the original PDEF file when used in conjunction with the *-new_cells_only* option of the *write_clusters* command.

Data Types

Boolean

Default true in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

When this variable is set to *true*, the *write_clusters -new_cells_only* command writes the new cells that are not in the original PDEF file. It determines the new cells by comparing the cell names.

When this variable is set to *false*, the *write_clusters -new_cells_only* command writes the cells that were added during the last run of the *reoptimize_design* command.

physopt_area_critical_range

Specifies a margin of slack for cells during area optimization. If a cell has a slack less than the area critical range, area optimization is not done for the cell.

Data Types

float

Default Variable is not supported in Design Compiler non-topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

-1.04858e+06 in Design Compiler topographical mode and Design Compiler NXT topographical mode

Description

The *physopt_area_critical_range* variable specifies a margin of slack for cells during area optimization. If a cell has a slack less than the area critical range, area optimization is not done for the cell. The default value is minus infinity; that is, all cells are optimized for area during area recovery.

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

---

physopt_create_missing_physical_libcells

Directs the tool to create dummy physical descriptions of missing physical library cells.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode and Design Compiler NXT topographical mode

Group

physopt

Description

This variable enables the tool to create a dummy physical cell if a cell is missing from the user-supplied physical libraries.

The tool uses a simple physical description for these dummy physical cells. This is mainly required in the exploration flow so that the tool does not stop when you do not have the final physical libraries. This is not intended as a replacement for specifying the correct physical libraries. For a final and accurate solution, you must specify the correct physical libraries.

The dummy physical cells created by the tool are not persistent. They exist only in the current session and are not stored in the database. These cells are recreated in a new session, if they are still required.

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

---

physopt_enable_root_via_res_support

Enable via resistance support in the command *compile_ultra -spg*.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Feedback

false in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Group

syn-qor

Description

Controls the support of via resistance.

The default value for this variable is false.

See Also

- compile_ultra

---

physopt_enable_via_res_support

Enables and disables the support of via resistance for RC estimation.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

true in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Controls the support of via resistance for RC estimation. To disable via resistance for RC estimation, set this variable to *false*.

To see the current value of this variable, use the *printvar physopt_enable_via_res_support* command.

---

physopt_hard_keepout_distance

Specifies the keepout distance used during virtual layout generation in the *compile_ultra* flow.

Data Types

float

Default Variable is not supported in Design Compiler non-topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

0 in Design Compiler topographical mode and Design Compiler NXT topographical mode

Description

You can use this feature in Design Compiler topographical mode. This variable specifies the keepout distance used during virtual layout generation in the *compile_ultra* flow. Specify the distance in micron units.

To prevent congestion, it is useful to mark an area in the design that surrounds a fixed macro as a hard keepout area. If you specify an area as a hard keepout area, the tool does not place any cells in that area. Define the area to be marked by setting a value for this variable, which the tool then uses to inflate the fixed cell's rectangle the same distance in all four directions.

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

See Also

- compile_ultra

---

physopt_ignore_lpin_fanout

Ignores max fanout constraints that are specified in the library.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Group

physopt

Description

This variable instructs tools to ignore max_fanout violations from the constraints in the library. Only optimization ignores those constraints and the *report_constraint* command still reports the violation.

See Also

- report_constraint

Feedback

physopt_power_critical_range

Specifies a margin of slack for cells during leakage power optimization. If a cell has a slack less than the power critical range, power optimization will not be done for the cell.

Data Types

float

Default Variable is not supported in Design Compiler non-topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

-1.04858e+06 in Design Compiler topographical mode and Design Compiler NXT topographical mode

Group

physopt

Description

This variable specifies a power critical range, which is a margin of slack for cells during leakage power optimization. If a cell has a slack less than the power critical range, leakage power optimization is not performed for the cell.

By default, the value of minus infinity indicates that all cells are optimized for leakage power during the leakage power optimization phase of the *physopt* command.

placer_auto_timing_control

Enables automatic timing control for direct timing-driven placement when you use the *compile_ultra -spg* and *compile_ultra -incremental -spg* commands.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

true in Design Compiler topographical mode and Design Compiler NXT topographical mode

Description

When you set this variable to *true*, the direct timing-driven placement tries to focus the timing optimization on the most critical timing paths to find a balance between reducing the worst negative slack (WNS) and reducing the total negative slack (TNS).

Feedback

This also enables Zroute-based congestion-driven placement instead of congestion estimation based on virtual routing, which estimates congestion based on wire-routing predictions.

You can use this feature in Design Compiler NXT topographical mode.

To use the automatic timing control feature, you must use the *set_icc2_options* command as follows:

```
prompt> set_app_var placer_auto_timing_control true
prompt> set_icc2_options -ref_lib "lib1.ndm lib2.ndm" -technology tech.tf
```

See Also

- set_icc2_options

---

placer_buffering_aware

Enables buffering-aware timing-driven placement when you use the *compile_ultra -spg* command.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

true in Design Compiler topographical mode and Design Compiler NXT topographical mode

Description

When you set this variable to *true*, the tool enables buffering-aware timing-driven placement by using an approximate timing model that estimates the effects of buffering long nets and high-fanout nets later in the flow. The netlist is not changed.

This feature is used during the initial placement to get a better starting point for later timing optimizations.

You can use this feature in Design Compiler NXT topographical mode.

To use the buffering-aware placement feature, you must use the *set_icc2_options* command as follows:

```
prompt> set_app_var placer_buffering_aware true
prompt> set_icc2_options -ref_lib "lib1.ndm lib2.ndm" -technology tech.tf
```

Feedback

See Also

- set_icc2_options

---

placer_channel_detect_mode

Controls the treatment of channel areas during coarse placement.

Data Types

string

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

This variable controls the treatment of channel areas during coarse placement.

Valid values are *auto*, *true*, or *false*.

By default (*false*), the tool does not treat channel areas differently.

When you set this variable to *true*, the tool analyzes the area available for placement and classifies narrow parts as channels. The tool reduces the maximum cell density allowed in channel areas during coarse placement to help reduce congestion.

When you set this variable to *auto*, the tool considers the amount of channel area in the design. If the tool determines that the design has little or no channel area, it does not treat channel areas differently. If the tool determines that the design has a substantial amount of channel area, it reduces the maximum cell density allowed in channel areas during coarse placement to help reduce congestion.

Some increase in runtime and memory usage is expected when this feature is active. Setting this variable to *auto* only incurs the runtime and memory penalties when the design has a substantial amount of channel area.

---

placer_cong_restruct

Enables congestion-driven restructuring when you use the *compile_ultra -spg* command

Data Types

Boolean

Feedback

Default Variable is not supported in Design Compiler non-topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode and Design Compiler NXT topographical mode

Description

When you set this variable to *true*, the tool enables congestion-driven restructuring when you use the *compile_ultra -spg* command. The tool performs restructuring of the net during the initial placement inside the wire-length-driven placement.

You can use this feature in Design Compiler NXT topographical mode.

To use the congestion-driven restructuring feature, you must use the *set_icc2_options* command as follows:

```
prompt> set_app_var placer_cong_restruct true
prompt> set_icc2_options -ref_lib "lib1.ndm lib2.ndm" -technology tech.tf
```

See Also

- set_icc2_options

---

placer_congestion_effort

Controls which congestion estimator to use during congestion-driven placement in the *place_opt -congestion* or *compile_ultra -spg* flow.

Data Types

string

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

auto in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

The *placer_congestion_effort* variable controls which congestion estimator to use during congestion-driven placement in the *place_opt -congestion* or *compile_ultra -spg* flow.

Valid values are *auto* or *medium*. For the *placer_congestion_effort* variable settings to take effect in the Design Compiler Graphical tool, you must first set the *placer_enable_enhanced_router* variable to *true*.

By default (*auto*), the tool first runs a fast internal estimator to determine how much congestion is there in the placement. If the tool determines that there is not much

Feedback

congestion, it uses the map with fast estimator to drive congestion-driven placement. If the tool determines that there is too much congestion, uses Zroute global router to do a more accurate congestion estimation and uses that map to drive congestion-driven placement.

When you set the variable to *medium*, the tool always uses the Zroute global router for congestion-driven placement.

To improve congestion correlation with the IC Compiler II tool, you should not use *auto* for 7 nm and lower-process nodes. With 7 nm and lower-process nodes designs, you should launch the IC Compiler II tool from the Design Compiler NXT tool using the *set_icc2_options -congestion_use_global_route* command for congestion estimation during placement.

See Also

- compile_ultra
- placer_enable_enhanced_router

---

placer_disable_auto_bound_for_gated_clock

Determines whether automatic group bounding is disabled for gated clocks created by Power Compiler.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

true in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

The gated-clock method reduces a design's power consumption by controlling the switching activity from the clock signal to flip-flop registers. The automatic bound function that this variable enables (when set to *false*, the default is true) works only for the gated-clock elements Power Compiler creates. The *placer_disable_auto_bound_for_gated_clock* variable does not affect your design unless you use the clock-gating functionality of Power Compiler.

To reduce clock skew, place the clock-gating element and the flip-flops it controls close together in your design. The tool automatically creates a group bound for each clock-gating element and the flip-flops the element drives. This group bound helps place the flip-flops closer to the clock-gating element.

Feedback

If this variable remains true, the tool does not create a group bound for the clock-gating elements created with Power Compiler.

See Also

- remove_bounds
- report_bounds
- create_bounds

placer_disable_macro_placement_timeout

Prevents the coarse placer from timing out during macro placement.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode and Design Compiler NXT topographical mode

Description

Use this variable to prevent the coarse placer from timing out during macro placement when running *create_placement* with floating macros.

By default, macro placement exits early if runtime is too long when compared to the rest of coarse placement. This is to prevent excessively long runtimes that can occur when there are too many floating macros, too many constraints on macros, or bad constraints. The result is that some macros may be placed illegally.

When this variable is set to true, the placer will let macro placement run to completion regardless of runtime.

The default value of this variable is false.

placer_dont_error_out_on_conflicting_bounds

Prevents the coarse placer from exiting with an error when it detects conflicting bounds.

Data Types

Boolean

Feedback

Default Variable is not supported in Design Compiler non-topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

true in Design Compiler topographical mode and Design Compiler NXT topographical mode

Description

Use this variable to prevent the tool from exiting with an error when it detects conflicting bounds when running *place_opt* or *create_placement*.

By default, the placer will automatically adjust the conflicting groupbounds and continue instead of exiting with an error.

When this variable is set to false, the placer checks for conflicting movebounds and groupbounds and errors out if a conflict is detected. These bounds can be user-specified bounds or bounds generated by the tool.

The default value of this variable is true.

See Also

- create_bounds

---

placer_enable_enhanced_router

Enables Zroute-based congestion-driven placement to perform more accurate, congestion-aware placement.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

When the *placer_enable_enhanced_router* variable is set to *true*, the tool uses Zroute-based congestion-driven placement instead of congestion estimation based on virtual routing, which estimates congestion based on wire-routing predictions. Zroute-based congestion estimation is enabled only on designs that meet the internal congestion threshold. Any designs that are not congested enough to meet the threshold are estimated by virtual routing (the default).

Feedback

Design Compiler Graphical and IC Compiler use the same internal congestion thresholds. As a result, Zroute-based congestion-driven placement provides better optimization and congestion correlation between Design Compiler Graphical and IC Compiler than virtual routing.

However, using Zroute congestion-driven placement might increase runtime and impact area and timing. Therefore, use it only on highly congested designs.

Note that the *set_congestion_options -layer* command is not supported when *placer_enable_enhanced_router* is enabled.

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

```
prompt> printvar placer_enable_enhanced_router
```

See Also

- set_congestion_options

---

placer_enable_enhanced_soft_blockages

Prevents cells already placed on soft blockages, at the beginning of placement, from being moved out of the soft blockages.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

true in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

This variable affects only incremental coarse placement. It has no effect on synthesis or legalization.

Soft blockages prevent the coarse placer from placing cells on the blockage. They do not prevent synthesis or legalization placing cells. After an initial placement, synthesis may add cells on top of a soft blockage. If incremental coarse placement is later done on the design, it will move those new cells off the soft blockages. This may be undesirable.

Set this variable to true to prevent incremental coarse placement from moving cells out of soft blockages.

If this variable is set to true, cells already on soft blockages at the beginning of incremental coarse placement will be allowed, but not constrained, to stay there.

Feedback

Cells not on soft blockages at the beginning of incremental coarse placement will always be prevented from moving onto the soft blockages during placement, irrespective of the value of this variable.

See Also

- create_placement_blockage

---

placer_enable_redefined_blockage_behavior

Enables new behavior and features related to placement blockages.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

true in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

This variable affects coarse placement when your design contains enhanced soft blockages, no-register blockages, buffer-only blockages, no-relative-placement blockages, and category blockages.

Set this variable to *true*, when:

- Your design contains any of the types of blockages previously listed, to prevent clumping of cells in the region outside the blockages.
- Your design contains more than one of the types of blockages previously listed, to ensure the correct interaction between the different types of blockages.
- You create buffer-only blockages using the *create_placement_blockage -buffer_only* command.
- You create category blockages using the *create_placement_blockage -category* command.

See Also

- create_placement_blockage

Feedback

placer_enhanced_low_power_effort

Controls Link Placer ELPP Effort Level.

This variable is supported only in Design Compiler NXT shell and in topographical mode.

Data Types

string

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

default in Design Compiler NXT topographical mode

Description

This variable controls the effort level of the "link placer enhanced low power placement" feature. This feature is only available when Total power optimization in DC-NXT is enabled and link placer is employed.

Allowed values are *none, low, medium, and high.*

When set to *none*, enhanced low power placement optimization is skipped for the link placer.

When set to *low*, ELPP effort for low power placement is low.

When set to *medium*, ELPP effort for low power placement is medium.

When set to *high*, ELPP effort for low power placement is high.

By default, low effort is used and medium/high efforts can be used to get more total power improvement.

To determine the current value of this variable, use the *printvar placer_enhanced_low_power_effort* command. For a list of all compile variables and their current values, use the *print_variable_group compile* command.

See Also

- compile
- compile_enable_total_power_optimization

placer_gated_register_area_multiplier

Specifies the value of the multiplier used to generate automatic group bounds for gated clocks.

Feedback

Data Types

float

Default Variable is not supported in Design Compiler non-topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

20 in Design Compiler topographical mode and Design Compiler NXT topographical mode

Description

The gated-clock method reduces a design's power consumption by controlling the switching activity from the clock signal to flip-flop registers. The automatic bound function works only for the gated-clock elements created by Power Compiler. The *placer_gated_register_area_multiplier* variable does not affect your design unless you use the clock-gating functionality of Power Compiler.

To reduce clock skew, place the clock-gating element and the flip-flops it controls close together in your design. The tool automatically creates a floating group bound for each clock-gating element and the flip-flops the element drives. This group bound helps place the flip-flops closer to the clock-gating element.

The size of the group bound is equal to the square root of the product of the total area of flip-flop elements the gate controls within the group bound and the value of the *placer_gated_register_area_multiplier* variable.

For example, if a clock-gating element drives five flip-flops and the total area of these six cells is 45, the size of the group bound is sqrt(20 x 45) = 30.

---

placer_max_allowed_timing_depth

Specifies the maximum timing graph depth allowed by the placer.

Data Types

integer

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

20000 in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

The placer cannot process extremely deep timing paths, and will check for the presence of such paths. This variable controls the depth limit for the checking.

When running the placer, no path may be deeper than this limit. This applies even if the path is unconstrained.

Feedback

If this variable is set above 32767, however, two things will happen: 1) Depth checking will be disabled. 2) Multicore operation will be disabled.

Be aware that increasing this limit may cause the process to abort due to insufficient stack space. Always set the process stack size to a high value in your system command shell when using this option.

An alternative to changing this limit is to use set_disable_timing to break very long paths into multiple segments. The paths must be broken up so than no segment (even if unconstrained) is longer than the limit.

---

placer_max_cell_density_threshold

Enables a mode of coarse placement in which cells can clump together.

Data Types

float

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

-1 in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

This variable enables a mode of coarse placement in which cells are not distributed evenly across the surface of the chip and are allowed to clump together.

The value you specify sets the threshold for how tightly the cells are allowed to clump. A value of 1.0 does not allow gaps between cells.

A reasonable value is one that is above the background utilization of your design but below 1.0. For example, if your background utilization is 40%, or 0.4, a reasonable value for this variable is a value between 0.4 and 1.0. The higher the value, the more tightly the cells clump together.

You can use this feature in Design Compiler topographical mode. It affects the virtual layout in the *compile_ultra* flow.

For example, to set the threshold to 0.7, enter

```
prompt> set placer_max_cell_density_threshold 0.7
```

See Also

- compile_ultra

Feedback

placer_max_parallel_computations

Overrides the value set by the *set_host_options -max_cores* command for running the placement engine in multicore mode.

Data Types

integer

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

0 in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Group

placer variables

Description

This variable can reduce the number of cores used for running the placement engine when you have previously enabled multicore processing by running the *set_host_options -max_cores* command.

If the variable is set to 0 (the default value), the maximum number of cores used by the placement engine is specified by the *set_host_options -max_cores* command.

If the variable is set to 1, the placement engine runs in single-core mode, regardless of the number of cores specified by the *set_host_options -max_cores* command.

If the variable is set to a value greater than 1 and smaller than the value specified by the *set_host_options -max_cores* command, the placement engine ignores the value set by the *set_host_options* command and runs in multicore mode with the number of cores specified by the variable.

If the variable is set to a value greater than the value specified by the *set_host_options -max_cores* command, this variable is ignored.

This variable should not be used for normal operation and is primarily intended for diagnostic purposes.

Examples

The following example enables multicore processing with a maximum of four cores for all multicore functions.

```
prompt> set_host_options -max_cores 4
```

The following example enables multicore processing with a maximum of two cores for the placement engine and a maximum of four cores for all other multicore functions.

Feedback

```
set_host_options -max_cores 4
set placer_max_parallel_computations 2
```

The following example enables singles-core processing for the placement engine and a maximum of four cores for all other multicore functions.

```
set_host_options -max_cores 4
set placer_max_parallel_computations 1
```

The following example enables multicore processing with a maximum of two cores for all multicore functions. The *placer_max_parallel_computations* variable is ignored because its value is greater than the value set the *set_host_options* command.

```
set_host_options -max_cores 2
set placer_max_parallel_computations 4
```

See Also

- set_host_options

---

placer_reduce_high_density_regions

Enables high-effort high-density-spot reduction during coarse placement.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

This variable enables high-effort high-density-spot reduction during coarse placement when set to *true*.

Use this variable on designs that exhibit high-density areas around macros or blockages when the default coarse placement effort is insufficient to remove all these high-density areas. You might see longer runtimes as a side effect (coarse placement runtime is expected to double). This feature is not yet wire-length or timing aware, so wire-length and timing might degrade as a result of using it.

To enable this feature, enter

```
prompt> set placer_reduce_high_density_regions true
```

Feedback

placer_show_zroutegr_output

Reports Zroute global routing information during congestion-driven placement.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode and Design Compiler NXT topographical mode

Description

This variable controls whether global routing information is reported during congestion-driven placement when the Zroute global router is used.

By default (*false*), the global routing information is not reported.

When you set this variable to *true*, the global routing information is reported when you use Zroute global routing for congestion-driven placement.

placer_tns_driven

Directs the timing-driven placer to optimize the total negative slack in the design instead of the worst negative slack during the initial compile.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode

true in Design Compiler NXT topographical mode

Description

When you set the placer_tns_driven variable to *true* before running the *compile_ultra* command, the placer tries to reduce the total negative slack of the design instead of optimizing the worst negative slack.

This variable is only supported in Design Compiler and Design Compiler NXT topographical modes. It is not supported in Design Compiler and Design Compiler NXT non-topographical modes or DC Explorer.

Feedback

In Design Compiler NXT topographical mode, the variable is on by default (*true*) and timing-driven placement enhanced to focus timing optimization on the most critical timing paths to find a good balance between reducing the worst slack and reducing the total negative slack.

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

```
prompt>printvar placer_tns_driven
```

See Also

- placer_tns_driven_in_incremental_compile

---

placer_tns_driven_in_incremental_compile

Controls how the tool optimizes the total negative slack in the design during incremental compile.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

auto in Design Compiler topographical mode and Design Compiler NXT topographical mode

Description

By default, the placer tries to reduce the total negative slack in the design only when the total negative slack exceeds a certain threshold.

To optimize the total negative slack in the design even if the total negative slack number is low, set the *placer_tns_driven_in_incremental_compile* variable to *true* before running the *compile_ultra -incremental* command.

To optimize the worst negative slack instead of the total negative slack, set the *placer_tns_driven_in_incremental_compile* variable to *false* before running the *compile_ultra -incremental* command.

This variable is only supported in Design Compiler topographical mode. It is not supported in Design Compiler non-topographical mode or DC Explorer.

Feedback

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

```
prompt>printvar placer_tns_driven_in_incremental_compile
```

See Also

- placer_tns_driven

---

placer_wide_cell_pg_strap_distance

Sets the power and ground (PG) pitch for the design to distribute wide cells evenly during coarse placement.

Data Types

float

Default Variable is not supported in Design Compiler non-topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

0 in Design Compiler topographical mode and Design Compiler NXT topographical mode

Description

This variable specifies the pitch of the PG straps in the design. You must use this variable along with the *placer_wide_cell_use_model* variable. These variables together enable a mode of coarse placement to distribute wide cells evenly across the surface of the chip to minimize displacement of cells during legalization and reduce congestion. Cells with a width greater than half the specified pitch are considered as wide cells.

You can use this feature in Design Compiler topographical mode.

For example, to specify the pitch of the PG straps,

```
prompt> set_app_var placer_wide_cell_use_model true
prompt> set_app_var placer_wide_cell_pg_strap_distance 0.3
```

See Also

- compile_ultra

---

placer_wide_cell_use_model

Enables wide cell support in the coarse placer.

Feedback

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode and Design Compiler NXT topographical mode

Description

This variable enables a mode of coarse placement to distribute wide cells evenly across the surface of the chip to minimize displacement of cells during legalization and reduce congestion. Cells with a width (of M1 PG pitch) greater than half the distance between adjacent M1 power and ground (PG) straps are considered as wide cells. You must use this variable along with the *placer_wide_cell_pg_strap_distance* variable.

You can use this feature in Design Compiler topographical mode.

For example, to improve the placement of wide cells in a region,

```
prompt> set_app_var placer_wide_cell_use_model true
```

See Also

- compile_ultra

---

port_complement_naming_style

Defines the convention the *compile* command uses to rename ports complemented as a result of using the *set_boundary_optimization* command.

Data Types

string

Default %s_BAR

Description

This variable defines the convention the *compile* command uses to rename ports complemented as a result of using the *set_boundary_optimization* command.

The variable string must contain one occurrence of %s (percent s). When *compile* generates a new port name, %s is replaced with the original name of the port. If this does not create a unique port name within the cell, the smallest possible integer that makes the name unique is appended to the end of the name (for example, X_BAR_0, X_BAR_1, and so on).

To determine the current value of this variable, use the *printvar port_complement_naming_style* command. For a list of all compile variables and their current values, use *print_variable_group compile*.

See Also

- compile
- set_boundary_optimization
- compile_disable_hierarchical_inverter_opt

---

power_cg_all_registers

Specifies to the *insert_clock_gating* command whether to clock gate all registers, including those that do not meet the necessary requirements.

Data Types

Boolean

Default false

Description

This variable specifies whether to clock gate all registers, including those that do not meet the necessary requirements. If this variable is set to *false* (the default value), registers that do not meet the setup, width, or enable condition are not considered for clock gating unless they are explicitly included with the *set_clock_gating_registers* command. When set to *true*, a redundant clock gate is inserted for these registers. This can be useful for clock tree balancing. If necessary, the redundant clock gates are duplicated to meet the max_fanout constraint. Note that the minimum_bitwidth constraint is not honored for the redundant clock gated registers.

To determine the current value of this variable, use the *printvar power_cg_all_registers* command. For a list of power variables and their current values, use *print_variable_group power*.

See Also

- elaborate
- insert_clock_gating

---

power_cg_auto_identify

Identifies clock-gating circuitry inserted by Power Compiler from a structural netlist.

Feedback

Data Types

Boolean

Default false

Description

When this variable is set to *true*, clock gates inserted by Power Compiler are automatically identified. Identification refers to the process of detecting clock gates and the corresponding gated element association, and setting different attributes on these objects. These attributes enable the subsequent steps in the tool to work efficiently. Set this variable with the *read_verilog* command before reading the input RTL file.

Clock gate identification is required when a netlist is passed in ASCII format. This is not required if a binary database (such as in .ddc format or MilkyWay) is used to transfer design data between tool invocations.

By using the automatic identification method, each time a command that works on clock-gating circuitry is called, identification is performed. This ensures that the command makes use of the current clock-gating configuration of the design.

In order to be identified, the gate clock must meet the following conditions:

- Be connected to a clock net through its clock pin
- All its gated registers are gated through the clock pin
- All its gated registers are activated by the same edge of the clock

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

See Also

- compile_ultra
- identify_clock_gating
- insert_clock_gating

---

power_cg_balance_stages

Controls whether gate stage balancing is on or off during *compile [-incremental_mapping] -gate_clock* or *compile_ultra [-incremental_mapping] -gate_clock*.

Data Types

Boolean

Feedback

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable reduces the unevenness in the number of stages of clock gates feeding different register banks. When this variable is set to *true*, the *compile -gate_clock* or *compile_ultra -gate_clock* command reconfigures the different stages of clock gates so that there is exactly the same number of clock gates in each path from the clock root to the clock pin of all gatable register banks. It is necessary to define clocks using the *create_clock* command for each clock network where clock gates need to be reconfigured.

Only the tool-inserted clock gates and integrated clock gating (ICG) cells are considered during stage balancing.

A clock-gating cell is not modified or removed if it or its parent hierarchical cell is marked as *dont_touch* with the *set_dont_touch* command; it will not be affected by clock gate stage balancing.

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

See Also

- compile
- compile_ultra
- report_clock_gating
- set_clock_gating_style
- power_cg_reconfig_stages

---

power_cg_cell_naming_style

Specifies the naming style for clock-gating cells created during *insert_clock_gating*.

Data Types

string

Default ""

Group

power

Feedback

Description

This variable determines the way that clock gating cells are named.

This variable must be set before issuing the *insert_clock_gating* command.

This variable can contain any string in addition to the tokens listed below. The tokens are replaced by the appropriate value during clock gating and other strings are retained without changes. For example:

```
set power_cg_cell_naming_style \\
   "prefix_%c_%e_midfix_%r_%R_%d_suffix"

  %c - clock name
  %n - immediate enable signal name
  %r - first gated register bank name (or module name for module
       clock gates and not applicable for factored clock gates)
  %R - all gated register banks sorted alphabetically (not applicable
       for module or factored clock gates)
  %d - index for splitting/name clash resolution
```

Only simple (non-hierarchical) names are used for all object names.

If there is no occurrence of "%d" in *power_cg_cell_naming_style*, the tool assumes a %d at the end. The following are examples of the variable usage:

```
prompt> set power_cg_cell_naming_style "clk_gate_%r_%d

prompt> set power_cg_cell_naming_style "clock_gate_cell_%c_%r_%d_name
```

See Also

- insert_clock_gating
- power_cg_gated_clock_net_naming_style
- power_cg_module_naming_style

---

power_cg_derive_related_clock

Derives the clock domain relationship between registers from the hierarchical context.

Data Types

Boolean

Default false

Feedback

Description

For latch-free clock gate insertion with the *insert_clock_gating* command, the setup condition must be met before a bank can be clock gated. This implies that the enable condition of the bank must be combinationally dependent on nets that are known to be synchronous (such as, belong to the same clock domain) with the registers of the bank that are being clock gated.

The clock domain of nets is determined by the registers withing the module (subdesign) that drive the nets and the clock relationship specified for ports with the *set_input_delay* command. By default, the clock domain relationship is analyzed within the module. This is to ensure that clock gating on subdesigns is context independent. If a module is instantiated from a design different than the current design, the clock-gating result will still be correct.

If the non-combinational driver (register or top-level port) of an input to a module exists outside of that module, the clock domain of that input is considered unknown and no latch-free clock gating can be performed with enable conditions that depend on that particular input.

The setup condition can be relaxed to perform context specific analysis of the clock domain relationship. This is achieved by setting the *power_cg_derive_related_clock* variable to *true*. In that case, the clock domain of any net will be derived from the non-combinational drivers in the (hierarchical) fanin of the net.

To determine the current value of this variable, use the *printvar power_cg_derive_related_clock* command. For a list of power variables and their current values, use *print_variable_group power*.

See Also

- insert_clock_gating
- set_clock_gating_style

---

power_cg_enable_alternative_algorithm

Specifies to the *insert_clock_gating*, *compile-gate_clock* and *compile_ultra -gate_clock* commands whether to use an alternative algorithm to find gatable registers.

Data Types

Boolean

Default false

Feedback

Description

Depending on the design, clock gating can have a long runtime. When set to *true*, *power_cg_enable_alternative_algorithm* changes the clock gating algorithm. The alternative algorithm that is enabled often reduces the runtime for clock gating, especially for larger designs, but might increase the number of clock gates that are created compared to the default algorithm. The number of gated registers may vary.

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

See Also

- compile
- compile_ultra
- insert_clock_gating

---

power_cg_ext_feedback_loop

Controls whether external feedback loops should be used to generate the enable condition for a register with its enable pin tied to logic 1. This variable affects only the *insert_clock_gating*, *compile -gate_clock*, and *compile_ultra -gate_clock* commands.

Data Types

Boolean

Default true

Description

This variable controls whether external feedback loops should be used to generate the enable condition for a register with its enable pin tied to logic 1.

By default (true), these registers are considered for clock gating by looking for an external feedback loop. If an external feedback loop is found, it is used to generated a valid enable condition for the register.

When this variable is set to false, registers with their enable pin tied to logic 1 are discarded as candidates to be clock gated.

Since this process is done only during the RTL clock gating stage, the effect of this variable is limited to the *insert_clock_gating* command or a full run of the *compile -gate_clock* or *compile_ultra -gate_clock* command. Note that this variable has no effect on incremental runs of the *compile* or *compile_ultra* command.

Feedback

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

See Also

- compile
- compile_ultra
- insert_clock_gating

---

power_cg_flatten

Specifies to different *ungroup* commands whether or not to flatten Synopsys clock-gating cells.

Data Types

Boolean

Default false

Description

This variable specifies to different *ungroup* commands whether or not to flatten Synopsys clock-gating cells. The list of commands include *ungroup*, *compile -ungroup*, *optimize_registers -ungroup*, *balance_registers*, and *ungroup*.

If this variable is set to *false* (the default value), the clock-gating cells are not flattened during any ungroup step. To flatten the clock gating cells, set the value to *true* before using the *ungroup* command or any other ungrouping steps listed above.

In normal usage, ungrouping the clock gates is not recommended. Ungrouping the clock gates before compile could have serious side effects. For example, ungrouped clock gates cannot be mapped to integrated clock-gating cells. Power Compiler commands such as *report_clock_gating*, *remove_clock_gating*, and *rewire_clock_gating* assume that the clock gates have a hierarchy of their own. Also, the tool's placement of clock gates and registers may not be optimal when clock gating cells are flattened. Flattened clock gates are supported when using integrated clock-gating cells, provided the flattening is done only after *compile*.

To determine the current value of this variable, use the *printvar power_cg_flatten* command. For a list of power variables and their current values, use *print_variable_group power*.

Feedback

See Also

- compile
- insert_clock_gating
- set_clock_gating_style
- ungroup

---

power_cg_gated_clock_net_naming_style

Specifies the naming style for gated clock nets created during *insert_clock_gating*.

Data Types

string

Default ""

Group

power

Description

This variable determines the way gated clock nets are named.

This variable must be set before issuing the *insert_clock_gating* command.

This variable can contain any string in addition to the different tokens listed below. The tokens are replaced by the appropriate value during clock gating. For example:

```
set power_cg_gated_clock_net_naming_style \\
   "prefix_%c_%n_%g_%d_suffix'

  %c - original clock
  %n - immediate enable signal name
  %g - clock gate (instance) name
  %d - index for splitting/name clash resolution
```

Only simple (non-hierarchical) names are used for all object names.

If there is no occurrence of "%d" in *power_cg_gated_clock_net_naming_style*, the tool assumes a %d at the end. The following is an example of the variable usage:

```
prompt> set power_cg_gated_clock_net_naming_style "gated_%c_%d"
```

Feedback

See Also

- insert_clock_gating
- power_cg_cell_naming_style
- power_cg_gated_clock_net_naming_style
- power_cg_module_naming_style

---

power_cg_high_effort_enable_fanin_analysis

Controls the execution of high effort analysis to rewire the fanin of gated cells during *compile_ultra [-incremental_mapping] -gate_clock*.

Data Types

Boolean

Default false

Description

This variable controls the effort level used by *compile_ultra [-incremental] -gate_clock* to rewire the fanin of gated cells. The high effort analysis analysis is only performed if the variable *power_cg_high_effort_enable_fanin_analysis* is set to *true*.

This analysis checks if the fanin of the data or enable pin of the gated register, or enable pin of the gated clock-gating cells, can be rewired. Rewiring will change the input logic of a register, or clock-gating cell, given that its gating cell is active. This effects the effort level to avoid having shared logic between the input pins of the registers, or clock-gating cells, and its gating cells. If this rewiring is not possible, then the input logic of the gated cells is not changed. When setting the effort level to high, more runtime is used when performing this analysis.

This analysis, regardless of effort level, will not:

- Cross hierarchical boundaries
- Affect shift registers or multibit registers

Please note that this analysis is only performed upon the gated cells.

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

Feedback

See Also

- compile
- compile_ultra
- report_clock_gating
- set_clock_gating_style

---

power_cg_ignore_setup_condition

Ignores the setup condition for latch-free clock gating.

Data Types

Boolean

Default false

Description

This variable ignores the setup condition for latch-free clock gating. For latch-free clock gate insertion with the *insert_clock_gating* command, the setup condition must be met before a bank can be clock gated. This implies that the enable condition of the bank must be combinationally dependent on nets that are known to be synchronous (belong to the same clock domain) with the registers of the bank that are being clock gated.

The setup condition can be ignored during latch-free clock gating. This is achieved by setting the *power_cg_ignore_setup_condition* variable to *true*. This is generally not safe, since latch-free clock gating of a bank with an enable that is not synchronous to the registers of that bank can lead to problems on the clock net. In this case it is the responsibility of the designer to ensure that the result after clock gating is functionally correct.

To determine the current value of this variable, use the *printvar power_cg_ignore_setup_condition* command. For a list of power variables and their current values, use *print_variable_group power*.

See Also

- insert_clock_gating
- set_clock_gating_style

Feedback

power_cg_inherit_timing_exceptions

Specifies that during *compile -gate_clock* or *compile_ultra [-incr] -gate_clock*, timing exceptions defined on registers must be automatically inferred onto the enable pin of the clock gate that is gating these registers.

Data Types

Boolean

Default false

Group

power

Description

If this variable is set before running the *compile -gate_clock* or *compile_ultra -gate_clock*, then clock gate insertion tries to inherit the timing exceptions of the gated registers to the clock gate enable pin. The following conditions apply:

- Only cell level exceptions are inherited. This means that all synchronous pins of a register must have the same set of timing exceptions before this can be inferred onto a clock gate.
- If a set of registers with the same enable conditions has different timing exceptions among them, then this is split into a smaller set of registers with homogeneous exceptions before clock gate insertion. This might create small banks that fall below the minimum bit width specified for clock gating and so they may remain ungated.

See Also

- compile
- compile_ultra

power_cg_iscgs_enable

Specifies the use of instance-specific clock-gating style for the *set_clock_gating_style* and *remove_clock_gating_style* commands, and clock gate insertion during compile.

Data Types

Boolean

Default false

Feedback

Description

This variable specifies whether or not to use instance-specific clock-gating style during clock gate insertion, as well as to enable commands or options related to instance-specific clock-gating style. When this variable is set to *true*, instance-specific clock-gating style is enabled.

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

See Also

- compile
- compile_ultra
- remove_clock_gating_style
- set_clock_gating_style

---

power_cg_module_naming_style

Specifies the naming style for clock gating modules created during *insert_clock_gating*.

Data Types

string

Default ""

Group

power

Description

This variable determines the way clock gate modules (hierarchical designs created for clock gates during *insert_clock_gating*) are named.

This variable must be set before issuing the *insert_clock_gating* command.

This variable can contain any string in addition to the different tokens listed below. The tokens are replaced by the appropriate value during clock gating, and prefix, midfix, and suffix are examples of any constant strings that can be specified.

```
set power_cg_module_naming_style \\
   "prefix_%e_%l_midfix_%p_%t_%d_suffix"

  %e - edge type (HIGH/LOW)
  %l - library name of ICG cell library (if using ICG cells) or
       concatenated target_library names
```

Feedback

```
%p - immediate parent module name
%t - top module (current design) name
%d - index used to resolve name clash
```

If there is no occurrence of "%d" in *power_cg_module_naming_style*, the tool assumes a %d at the end. The following are examples of the variable usage:

```
prompt> set power_cg_module_naming_style "SNPS_CLOCK_GATE_%e_%p

prompt> set power_cg_module_naming_style "clock_gate_module_%e_%t_%d
```

See Also

- insert_clock_gating
- power_cg_cell_naming_style
- power_cg_gated_clock_net_naming_style

---

power_cg_permit_opposite_edge_icg

Allows negedge ICGs to gate posedge registers, and posedge ICGs to gate negedge registers during *compile [-incremental_mapping] -gate_clock* or *compile_ultra [-incremental_mapping] -gate_clock*.

Data Types

Boolean

Default false

Description

Allows negedge ICGs to gate posedge registers, and posedge ICGs to gate negedge registers during *compile_ultra -gate_clock* or *compile -gate_clock*. You can only use this feature if the *power_cg_permit_opposite_edge_icg* variable is set to *true* before using the *set_clock_gating_style* command.

This feature is useful when you are restricted from using either posedge or negedge ICGs. Cell selection might be restricted because the appropriate cell is not in your target library or your cells are constrained by a dont_use attribute.

This transformation is ensured by adding an inverter in the fan-in and another one in the fan-out of the ICG.

See Also

- compile
- compile_ultra

Feedback

- report_clock_gating
- set_clock_gating_style
- set_dont_use

---

power_cg_physically_aware_cg

Enables the restructuring of the integrated clock-gating cell and the registers in the Synopsys physical guidance flow to provide better convergence on clock gating timing paths against postCTS results.

Data Types

Boolean

Default false

Description

Set this variable to true to enable the restructuring of the integrated clock-gating cell and the registers in the Synopsys physical guidance flow to provide better convergence on clock gating timing paths against postCTS results. Enabling this setting can degrade power and is hence recommended only if clock gating timing paths are expected to become critical during CTS.

The tool restructures the connection between the integrated clock-gating cell and the registers that it drives so that each clock-gating cell is physically close to the registers gated by the clock-gating cell.

Right before the restructuring process, and independent of the current setting of the variable *power_cg_auto_identify*, clock gating identification is performed.

Clock gates with high fanouts are split according to physical location of their gated registers. This placement is maintained throughout the implementation flow.

During the split operation, the user defined integrated clock-gating cell, specified by the *set_clock_gating_style* command, could be replaced by a better cell based on drive-strength characteristics. To prevent the using specific integrated clock-gating exclude them with the *set_dont_use -power* command.

To determine the current value of this variable, use the *printvar power_cg_physically_aware_cg* command. For a list of power variables and their current values, use the *print_variable_group power* command.

When this variable is enabled, the annotation of latency values from the *set_clock_gate_latency* command is disabled and the *apply_clock_gate_latency* command is also disabled.

Feedback

See Also

- apply_clock_gate_latency
- compile_ultra
- set_clock_gate_latency

---

power_cg_print_enable_conditions

Reports the enable conditions of registers and clock gates during clock gate insertion.

Data Types

Boolean

Default false

Description

This variable prints the enable conditions of clock gates and registers when performing automatic RTL clock gate insertion with the *insert_clock_gating* command. The report is useful for debugging purposes and to achieve a better understanding of the design structure and functionality.

The enable condition of a register or clock gate is a combinational function of nets in the design. As such, enable conditions can be represented by Boolean expressions of nets. The enable condition of a register represents the states for which a clock signal must be passed to the register. The enable condition of a clock gate corresponds to the states for which a clock will be passed to the registers in the fanout of the clock gate. The tool utilizes the enable condition of the registers for clock gate insertion.

To enable reporting of the enable conditions during clock gating, set *power_cg_print_enable_conditions* to *true* before issuing the *insert_clock_gating* command.

To determine the current value of this variable, use the *printvar power_cg_print_enable_conditions* command. For a list of power variables and their current values, use *print_variable_group power*.

See Also

- insert_clock_gating
- power_cg_print_enable_conditions_max_terms

Feedback

power_cg_print_enable_conditions_max_terms

Specifies the maximum number of product terms to be reported in the sum of product expansion of the enable condition.

Data Types

integer

Default 10

Description

This variable specifies the maximum number of product terms to be reported in the sum of product expansion of the enable condition. For debugging purposes and to achieve a better understanding of the design structure and functionality it can be useful to print the enable conditions of clock gates and registers when performing automatic RTL clock gate insertion with the *insert_clock_gating* command.

The enable conditions are reported as a sum of product expression. Since for complex expressions such a representation is known to grow very large, the maximum number of product terms is limited by the *power_cg_print_enable_conditions_max_terms* variable. By default, at most 10 product terms are printed. If the actual number of product terms exceeds this limit, the enable condition is reported as "?? (too many product terms)". If necessary, the number of product terms to be shown can be increased by setting *power_cg_print_enable_conditions_max_terms* to a larger value.

To determine the current value of this variable, use the *printvar power_cg_print_enable_conditions_max_terms* command. For a list of power variables and their current values, use *print_variable_group power*.

See Also

- insert_clock_gating
- power_cg_print_enable_conditions

power_cg_reconfig_stages

Controls the reconfiguration of multistage clock gates during *compile [-incremental_mapping] -gate_clock* or *compile_ultra [-incremental_mapping] -gate_clock*.

Data Types

Boolean

Feedback

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable controls whether reconfigurations are performed during *compile_ultra -gate_clock* or *compile -gate_clock*. The reconfigurations are performed only if the *power_cg_reconfig_stages* variable is set to *true*.

Multistage reconfiguration involves the addition or reduction of stages of clock gates. You specify the maximum permissible number of stages of clock gates in the design using the *set_clock_gating_style -num_stages* command. A clock gate stage is added when a common enable condition can be factored out of many clock gates without violating the maximum stages specified. A clock gate stage is reduced if the design has more stages than what is specified.

Also, it is necessary to define clocks using the *create_clock* command for each clock network where clock gates need to be reconfigured.

Only the tool-inserted clock gates and integrated clock-gating (ICG) cells can be reconfigured.

A clock-gating cell is not modified or removed if it or its parent hierarchical cell is marked *dont_touch* with the *set_dont_touch* command; it will not be affected by multistage reconfiguration.

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

See Also

- compile
- compile_ultra
- report_clock_gating
- set_clock_gating_style
- power_cg_balance_stages

power_clock_network_include_register_clock_pin_power

Indicates whether the register clock pin power is included when reporting *clock_network* power.

Feedback

Data Types

Boolean

Default true

Description

This variable affects the power report for the predefined *clock_network* and *register* power groups. When set to *true*, the internal power of registers caused by the toggling of register clock pin is included as *clock_network* power and excluded from *register* power. When set to *false*, the power is included as *register* power and excluded from *clock_network* power.

See Also

- report_power

---

power_default_static_probability

Specifies the default static probability value.

Data Types

float

Default 0.5

Description

The following variables are used to determine the switching activity of non-user annotated nets that are driven by primary inputs or black box cells:

```
        power_default_static_probability
power_default_toggle_rate
power_default_toggle_rate_type
```

For other unannotated nets, the tool propagates the switching activities of the driving cell inputs based on the cell functionality to derive the switching activity required for power calculations. This mechanism cannot be used for primary inputs and black box outputs. Instead, the following values are used for these types of nets:

- User annotated values are used, even when the net is partially annotated (for example, the static probability is annotated, but the toggle rate is not).
- In some cases, unannotated switching activity values may still be accurately derived; for example, if the net drives a buffer cell and the output of this cell is user annotated, then the user annotated values are used as the default values. Also, if the input is a

Feedback

clock, then the clock period and waveform are used to derive the switching activity values.

- If the static probability is not annotated, then the value of the *power_default_static_probability* variable is used for the static probability value.
- If the toggle rate is not annotated, then the default toggle rate value is derived from the *power_default_toggle_rate_type* and *power_default_toggle_rate* values. If the value of the *power_default_toggle_rate_type* variable is *fastest_clock* then the toggle rate value is

  ```
  dtr * fclk
  ```

  where fclk is the frequency of the related clock if specified by the *set_switching_activity* command, or the frequency of the fastest clock in the design. If the design has no clocks, then a value of 1.0 is used for fclk, and dtr is the value of the *power_default_toggle_rate* variable.
- If the value of *power_default_toggle_rate_type* is *absolute*, then the value of the *power_default_toggle_rate* variable is used as the toggle rate.

The value of *power_default_static_probability* must be between 0.0 and 1.0, both inclusive. The value of *power_default_toggle_rate* must be greater or equal to 0.0. Also, if the value of *power_default_static_probability* is 0.0 or 1.0, then the value of *power_default_toggle_rate* must be 0.0. If the value of *power_default_toggle_rate* is 0.0, then the value of *power_default_static_probability* must be either 0.0 or 1.0.

The default value of *power_default_toggle_rate* is 0.1. The default value of *power_default_static_probability* is 0.5. The value of *power_default_toggle_rate_type* can be either *fastest_clock* or *absolute*. The default value is *fastest_clock*.

See Also

- set_switching_activity
- power_default_toggle_rate
- power_default_toggle_rate_type

---

power_default_toggle_rate

Specifies the default toggle rate value.

Data Types

float

Default 0.1

Feedback

Description

The following variables are used to determine the switching activity of non-user annotated nets that are driven by primary inputs or black box cells:

```
        power_default_static_probability
power_default_toggle_rate
power_default_toggle_rate_type
```

For other unannotated nets, the tool propagates the switching activities of the driving cell inputs based on the cell functionality to derive the switching activity required for power calculations. This mechanism cannot be used for primary inputs and black box outputs. Instead, the following values are used for these types of nets:

- User annotated values are used, even when the net is partially annotated (for example, the static probability is annotated, but the toggle rate is not).
- In some cases, unannotated switching activity values may still be accurately derived; for example, if the net drives a buffer cell and the output of this cell is user annotated, then the user annotated values are used as the default values. Also, if the input is a clock, then the clock period and waveform are used to derive the switching activity values.
- If the static probability is not annotated then the value of the *power_default_static_probability* variable is used for the static probability value.
- If the toggle rate is not annotated, then the default toggle rate value is derived from the *power_default_toggle_rate_type* and *power_default_toggle_rate* values. If the value of the *power_default_toggle_rate_type* variable is *fastest_clock* then the following is used for the toggle rate value:

  dtr * fclk

  where fclk is the frequency of the related clock if specified by the *set_switching_activity* command, or the frequency of the fastest clock in the design. If the design has no clocks, then a value of 1.0 is used for fclk, and dtr is the value of the *power_default_toggle_rate* variable.
- If the value of *power_default_toggle_rate_type* is *absolute*, then the value of the *power_default_toggle_rate* variable is used as the toggle rate.

The value of *power_default_static_probability* must be between 0.0 and 1.0, both inclusive. The value of *power_default_toggle_rate* must be greater or equal to 0.0. Also, if the value of *power_default_static_probability* is 0.0 or 1.0, then the value of *power_default_toggle_rate* must be 0.0. If the value of *power_default_toggle_rate* is 0.0, then the value of *power_default_static_probability* must be either 0.0 or 1.0.

Feedback

The default value of *power_default_toggle_rate* is 0.1. The default value of *power_default_static_probability* is 0.5. The value of *power_default_toggle_rate_type* can be either *fastest_clock* or *absolute*. The default value is *fastest_clock*.

See Also

- set_switching_activity
- power_default_static_probability
- power_default_toggle_rate_type

---

power_default_toggle_rate_type

Specifies the default toggle rate type.

Data Types

string

Default fastest_clock

Description

The following variables are used to determine the switching activity of non-user annotated nets that are driven by primary inputs or black box cells:

```
power_default_static_probability
power_default_toggle_rate
power_default_toggle_rate_type
```

For other unannotated nets, the tool propagates the switching activities of the driving cell inputs based on the cell functionality to derive the switching activity required for power calculations. This mechanism cannot be used for primary inputs and black box outputs. Instead, the following values are used for these types of nets:

- User annotated values are used, even when the net is partially annotated (for example, the static probability is annotated, but the toggle rate is not).
- In some cases, unannotated switching activity values may still be accurately derived; for example, if the net drives a buffer cell and the output of this cell is user annotated, then the user annotated values are used as the default values. Also, if the input is a clock, then the clock period and waveform are used to derive the switching activity values.
- If the static probability is not annotated, then the value of the *power_default_static_probability* variable is used for the static probability value.
- If the toggle rate is not annotated, then the default toggle rate value is derived from the *power_default_toggle_rate_type* and *power_default_toggle_rate* values. If the value of

Feedback

the *power_default_toggle_rate_type* variable is *fastest_clock*, then the following is used for the toggle rate value:

```
dtr * fclk
```

where fclk is the frequency of the related clock if specified by the *set_switching_activity* command, or the frequency of the fastest clock in the design. If the design has no clocks, then a value of 1.0 is used for fclk, and dtr is the value of the *power_default_toggle_rate* variable.

- If the value of *power_default_toggle_rate_type* is *absolute*, then the value of the *power_default_toggle_rate* variable is used as the toggle rate.

The value of *power_default_static_probability* must be between 0.0 and 1.0, both inclusive. The value of *power_default_toggle_rate* must be greater or equal to 0.0. Also, if the value of *power_default_static_probability* is 0.0 or 1.0, then the value of *power_default_toggle_rate* must be 0.0. If the value of *power_default_toggle_rate* is 0.0, then the value of *power_default_static_probability* must be either 0.0 or 1.0.

The default value of power_default_toggle_rate variable is 0.1. The default value of power_default_static_probability variable is 0.5. The value of *power_default_toggle_rate_type* can be either *fastest_clock* or *absolute*. The default value is *fastest_clock*.

See Also

- set_switching_activity
- power_default_static_probability
- power_default_toggle_rate

---

power_derive_rtl_saif_map

Allows to derive names from RTL for the mapping file so it can be generated without reading a SAIF file.

Data Types

Boolean

Default false

Description

When setting this variable to true, the tool automatically derives RTL names based on the elaborated netlist to create a PrimePower or IC Compiler mapping file. Therefore, a mapping file can be generated without reading a SAIF file using the -auto_map_names

option in read_saif command, reading the SAIF file using saif_map -create_map or using saif_map -add_name to set manual RTL names on objects.

The derivation of the RTL names occurs during the writing of the mapping file, therefore it is not possible to obtain the derived names using saif_map -get_object_names.

The derivation of RTL names occurs only for registers, pre-existing ICGs, black boxes, memories and primary input ports.

Examples

Use the following flow to generate a PrimePower or IC Compiler II mapping file without reading a SAIF file:

```
prompt> set_app_var hdlin_enable_upf_compatible_naming true
prompt> set_app_var power_derive_rtl_saif_map true
prompt> saif_map -start
prompt> analyze
prompt> elaborate top
prompt> current_design top
prompt> compile_ultra
prompt> saif_map -write_map dc.ptpx -type ptpx
prompt> saif_map -write_map dc.icc2.map
```

---

power_do_not_size_icg_cells

Controls whether *compile* does not size the integrated clock-gating cells in a design to correct DRC violations because doing so may result in lower area and power.

Data Types

Boolean

Default true in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

When this variable is set to *true*, *compile* does not size the integrated clock-gating cells in the design to correct DRC violations, because doing so may result in lower area and power when the integrated clock-gating cell is the last element in the clock tree and drives all gated registers.

If the clock tree synthesis (CTS) tool inserts buffers after the clock gating, the fanout of the integrated clock-gating cell is limited to the clock-tree buffers. While running *compile* before performing clock tree synthesis, this information is not available in the design. If you set *power_do_not_size_icg_cells* to *true*, *compile* ignores DRC violations for the integrated

Feedback

clock-gating cells, because the CTS tool would insert buffers at the output of the cell. Once your design netlist has CTS buffers, you can set this variable to *false* to enable *compile* to fix any DRC violation still existing for the integrated clock-gating cell.

To determine the current value of this variable, use the *printvar power_do_not_size_icg_cells* command. For a list of power variables and their current values, use *print_variable_group power*.

See Also

- compile
- elaborate
- set_clock_gating_style

---

power_enable_clock_scaling

Enables or disables clock scaling for power analysis in Design Compiler.

Data Types

Boolean

Default false

Description

When set to *true*, this variable enables Design Compiler to scale dynamic power number according to clock frequencies specified in the SDC and the *set_power_clock_scaling* command.

For the current value of this variable, type the following command:

```
printvar power_enable_clock_scaling
```

See Also

- set_power_clock_scaling
- printvar

---

power_enable_datapath_gating

Enables or disables datapath gating technique for a design.

Feedback

Data Types

Boolean

Default true in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable enables or disables datapath gating, which is a dynamic power optimization technique for a design. By default, this variable is *true* which enables datapath gating. To disable datapath gating, set this variable to *false*.

Datapath gating inserts isolation cells for operators that have Synopsys low-power DesignWare architectures present in the synthetic library files. Datapath gating is also performed on all datapath blocks extracted during the High Level Optimization step. For datapath gating, the tool always uses the adaptive gating style.

This control only takes effect when minpower optimization flow is turned on. The minpower optimization is controlled by the *power_enable_minpower* variable, whose default value is *false*. To enable the datapath gating, both *power_enable_minpower* and *power_enable_datapath_gating* variables should be set to *true*. See *power_enable_minpower* for details.

To determine the current value of this variable, do the following:

```
prompt> printvar power_enable_datapath_gating.
```

For a list of power variables and their current values, type *print_variable_group power*.

See Also

- set_datapath_gating_options
- power_enable_minpower

---

power_enable_minpower

Enable/disable minpower optimization flow.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Feedback

Variable is not supported in DC Explorer

Description

This variable can be used to enable or disable minpower optimization flow, which will leverage many power optimization techniques or use power optimized implemenations for syntehtic parts. By default this variable is *false* which disables minpower flow. To enable minpower flow, set this variable to *true*.

Historically minpower optimization flow is controlled by inserting or removing *dw_minpower.sldb* to/from *synthetic_library* and *link_library*. Since 2019.03 release this control is replaced by *power_enable_minpower* variable.

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

See Also

- power_enable_datapath_gating

---

power_enable_one_pass_power_gating

Enables the one-pass flow power gating. This variable is for use only in non-UPF mode.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable enables one-pass flow power gating when set to *true*. One-pass flow eliminates the need for an incremental compile, and simplifies the user interface.

The following are examples of the original power-gating flow and the one-pass flow:

• Original power-gating flow:

```
prompt> set_power_gating_style -type CLK_FREE \\
        [get_cells lev1b_inst/*reg*]

prompt> set_power_gating_signal -power_pin_index 1 \\
        [get_pin lev1b_inst/retain]

prompt> set_power_gating_signal -power_pin_index 2 \\
```

Feedback

```
        [get_pin lev1b_inst/shutdown]

prompt> set power_enable_power_gating true

prompt> compile

prompt> hookup_power_gating_ports

prompt> compile -incr

prompt> write -format verilog -hierarchy -output post_compile.v
```

- One-pass power-gating flow:

```
prompt> set_power_gating_style -type CLK_FREE \\
        [get_cells lev1b_inst/*reg*]

prompt> set_power_gating_signal -power_pin_index 1 \\
        [get_pin lev1b_inst/retain]

prompt> set_power_gating_signal -power_pin_index 2 \\
        [get_pin lev1b_inst/shutdown]

prompt> set power_enable_one_pass_power_gating true

prompt> hookup_power_gating_ports -port_naming_style \\
        {lev1a_inst/retain lev1a_inst/shutdown} \\
        -default_port_naming_style power_pin_%d

prompt> compile

prompt> write -format verilog -hierarchy -output post_compile.v
```

power_enable_power_gating

Enables the power-gating flow that allows the selected retention registers from the target library to be used to map sequential elements. This variable can be used only in non-UPF mode. In UPF mode, use UPF commands to enable the power-gating flow.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Feedback

Group

power

Description

The *power_enable_power_gating* variable enables and disables the power gating flow during *compile* and *physopt* command activity.

Retention registers are the registers that can save the values of the registers and restore them later. Retention registers might have different styles. The *power_gating_cell* cell-level attributes in target libraries are used to specify the styles.

To allow only the retention registers with certain types used for the specified sequential elements during *compile*, set the *power_enable_power_gating* variable to *true*.

The command requires a Power Compiler license during *compile*, otherwise Design Compiler cannot handle retention registers correctly. If the variable is *true*, it also prevents the *compile* and *physopt* commands from swapping the retention registers back to regular sequential elements.

See Also

- compile

---

power_fix_sdpd_annotation

Specifies whether user-annotated SDPD switching activity annotation is corrected before it is used.

Data Types

Boolean

Default true in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable specifies whether the tool modifies the user-annotated state-dependent and/or path-dependent (SDPD) switching activity to fix any inconsistencies before they are used. The accuracy of switching activity annotation, including the SDPD annotation, affects the accuracy of power calculation, and when this variable is set the tool checks the SDPD annotation for inconsistencies. The SDPD annotation is automatically modified to fix any inconsistencies found.

Feedback

In most cases, some inconsistencies are created during normal switching activity flows, and the SDPD fixing step modifies the SDPD annotation slightly to improve the power estimation accuracy. For example, during SAIF generation using a simulator, the total of rise and fall toggle on a pin may be different, when perhaps an odd number of toggles are captured. In this case, the SDPD fixing step scales the SDPD toggle rate annotations so that the rise and fall totals are the same.

Setting this variable to *false* disables the SDPD fixing step.

Setting the *power_fix_sdpd_annotation_verbose* variable to *true* makes the SDPD fixing step issue verbose messages when user annotated switching activity is modified.

The following checks and modifications are performed during the SDPD fixing step:

- False states (states that always evaluate to 0, including the default state on cells whose other states cover all possible states) with non-zero state-dependent (SD) static probabilities have their static probability set to 0.0.
- Unannotated false states on cells with partially annotated SD static probabilities are automatically annotated with 0.0.
- Cells with fully annotated SD static probabilities have their static probabilities scaled so that they add up to 1.0.
- Cells with partially annotated SD static probabilities that add up to more than 1.0 have their static probabilities scaled down so that they add up to 1.0. On such cells, an SD static probability of 0.0 is set on the unannotated states.
- False states and arcs with non-zero state-dependent and/or path-dependent (SDPD) toggle rates have their toggle rate set to 0.0.
- Unannotated false states/arcs on pins with partially annotated SDPD toggle rates are automatically annotated with 0.0.
- Pins with fully annotated SDPD toggle rates have their toggle rates scaled so that they add up to the non-SDPD toggle rate of the pin (that is, the non-SDPD toggle rate on the net connected to the pin).
- Pins with partially annotated SDPD toggle rates that add up to more than the pin toggle rate have their SDPD toggle rates scaled down so that they add up to the non-SDPD toggle rate. On such pins, an SDPD toggle rate of 0.0 is set on the unannotated states and arcs.
- Pins with partially annotated SDPD toggle rates have their toggle rates scaled so that the totals of rise and fall SDPD toggle rates are equal.
- The sum of the SDPD toggle rates on pins with fully annotated SDPD toggle rate information is annotated as the non-SDPD pin toggle rate (that is, the total toggle rate of the net connected to the pin) if this was not previously annotated.

Feedback

See Also

- power_fix_sdpd_annotation_verbose
- power_sdpd_message_tolerance

---

power_fix_sdpd_annotation_verbose

Specifies whether verbose messages are reported during the fixing of user-annotated SDPD switching activity.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

When this variable is set to *true*, an information or warning message is reported for every modification (exceeding a tolerance criteria) to the user-annotated SDPD switching activity performed by the SDPD fixing step. The SDPD fixing step is enabled by the *power_fix_sdpd_annotation* variable. See the man page for *power_fix_sdpd_annotation* for more information on this step.

By default, the *power_fix_sdpd_annotation_verbose* verbose messages are not reported and a single message indicating that SDPD annotation is being modified is reported instead. Whether this variable is set to *true* or not, SDPD fixing messages are only reported if they exceed the tolerance criteria specified by the *power_sdpd_message_tolerance* variable.

See Also

- power_fix_sdpd_annotation
- power_sdpd_message_tolerance

---

power_handle_clock_network_power_as_ideal

Forces the tool to implicitly handle clock network power as ideal at pre-CTS stages.

Data Types

Boolean

Feedback

Default false

Description

The *power_handle_clock_network_power_as_ideal* variable forces the tool to implicitly handle clock network power as ideal at pre-CTS stages. When this variable is set to true, for clock network, total output net capacitance will be treated as zero and input net transition will be ideal for internal power calculation; total net capacitance will be treated as zero for switching power calculation.

The *power_handle_clock_network_power_as_ideal* variable is off (false) by default.

See Also

- report_power

---

power_hdlc_do_not_split_cg_cells

Specifies that the *insert_clock_gating* command will not split clock-gating cells to limit their fanout.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable, when set to *true* specifies that the *insert_clock_gating* command will not split clock-gating cells to limit their fanout. By default, *insert_clock_gating* splits clock-gating cells to limit their fanout. This activity is based on the value specified by the *set_clock_gating_style -max_fanout* command, whose default is unlimited. When *true*, *insert_clock_gating* does not split clock-gating cells, resulting in a netlist where all registers are gated by a single clock-gating cell if they share the same enable signal. Also, *insert_clock_gating* does not honor the value specified by *set_clock_gating_style -max_fanout*.

Set the variable to *true* if your clock-tree synthesis (CTS) tool inserts buffers after the clock-gating cell. In this case, the fanout of the clock-gating cell is limited to the buffers inserted by the CTS tool. This information is not available in the design while the *insert_clock_gating* command is performing RTL clock gating. When you set this variable to *true*, *insert_clock_gating* does not split the load of the clock-gating cells (by duplicating the cells) to save area and power.

Feedback

To determine the current value of this variable, use the *printvar power_hdlc_do_not_split_cg_cells* command. For a list of power variables and their current values, use *print_variable_group power*.

See Also

- compile
- insert_clock_gating
- set_clock_gating_style

---

power_keep_license_after_power_commands

Affects the amount of time a Power Compiler license is checked out during a shell session.

Data Types

Boolean

Default false

Description

This variable affects the amount of time a Power Compiler license is checked out during a shell session.

When set to *true*, a Power Compiler license that is checked out remains checked out throughout the shell session. When this variable is set to *false* (the default value), the Power Compiler license remains checked out only as long as a command is using it, and at the completion of the command, the license is released.

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

See Also

- report_power

---

power_lib2saif_rise_fall_pd

Specifies whether the *lib2saif* generates forward SAIF files with directives to generate rise and fall dependent path-dependent toggle counts.

Data Types

Boolean

Feedback

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

When this variable is set to *true*, the *lib2saif* command generates forward SAIF files with directives to generate separate rise and fall values for non-state-dependent path-dependent toggle counts.

When this variable is set to *false*, directives to generate just the total of the rise and fall values for non-state-dependent path-dependent toggle counts are generated.

For more accurate power calculations, use separate rise and fall toggle counts; however, older simulators and simulation interfaces may not recognize such directives and will fail to read the library forward SAIF file. The Synopsys SAIF generation PLI utility provided with Power Compiler X-2005.09 supports directives for separate rise and fall values for path-dependent toggle rates, but older versions of the utility do not.

Note that this variable affects only directives for path-dependent toggle counts that are not state-dependent. Whether this variable is set to true or false, directives for separate rise and fall values are generated for state-dependent and both state and path-dependent toggle counts.

See Also

- lib2saif

---

power_low_power_placement

Enables low-power placement during compile, which places cells closer together when they have high switching activity.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable, when set to *true*, enables low-power placement during a compile operation using the *compile_ultra -spg* command. Low-power placement assigns a higher cost to

Feedback

nets with high switching activity, so that the placer places cells closer together when they are connected by such nets. This tends to reduce the total switching power of the design.

To be effective, you should annotate switching activity information on the netlist using the *read_saif* command before the compile operation. If the netlist is partially annotated, the tool performs switching activity propagation to estimate switching activity on all nets and pins of the design, before low-power placement.

Low-power placement is a dynamic power reduction technique that uses advanced placement features only available in the *-spg* mode of the *compile_ultra* command. To enable and use low-power placement:

- Set the *power_low_power_placement* variable to *true*
- Enable dynamic power optimization. In a multicorner-multimode design, use the *set_scenario_options -dynamic_power true* command. In a non-multicorner-multimode design, use *set_dynamic_optimization true* command.
- Use physical guidance during optimization, *compile_ultra -spg*

See Also

- set_dynamic_optimization
- set_scenario_options
- compile_ultra

---

power_min_internal_power_threshold

Specifies the minimum cell internal power value that can be used in power calculations.

Data Types

string

Default "" in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

The value of this variable specifies the minimum threshold used for the cell internal power value. Internal power values are computed using the cell's switching activity and internal power characterization. If this variable has a numeric value and a cell's computed internal power is less than the variable's value, then the variable's value is used instead.

This variable has an effect only if it has a numeric value. The default value of this variable is "", which is non-numeric and so specifies that no minimum threshold is used on the cell internal power.

In general, this variable should not be used, since the internal power characterization specifies the correct internal power values.

See Also

- report_power

---

power_model_preference

Specifies the preference between the CCS power and the NLPM models in library cells that have power specified in both models.

Data Types

string

Default nlpm in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

A library can contain CCS power, NLPM, or both types of data within a cell definition. This variable specifies the power model preference if the library contains both NLPM and CCS power data.

Allowed values are as follows:

ccs instructs Power Compiler to use CCS power data in the library (if present) to calculate both static and dynamic power. If CCS power data is not found, Power Compiler uses NLPM data.

nlpm (the default) instructs Power Compiler to use NLPM data. If NLPM data is not found, Power Compiler uses CCS power data.

If neither CCS power nor NLPM data is found for a cell in the library, this cell is not characterized for power analysis.

See Also

- report_power

Feedback

power_rclock_inputs_use_clocks_fanout

Specifies whether clock network objects in an input port fanout are used to infer the input port's related clock.

Data Types

Boolean

Default true in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable is used during related clock inference to decide whether the inferred related clock on a design port is chosen to be the fastest clock whose clock network objects are in the port's transitive fanout. For example, when this variable is set to *true* (the default), if the transitive fanout of an input port contains a number of cells, the fastest clock on these flip-flop cells is chosen as the inferred related clock on the input port. When the variable is set to *false*, the input port will not have an inferred related clock.

For more information on the mechanism used to infer related clock information, see the *propagate_switching_activity* command man page.

See Also

- propagate_switching_activity
- power_rclock_unrelated_use_fastest
- power_rclock_use_asynch_inputs

power_rclock_unrelated_use_fastest

Specifies whether the fastest clock is set as the related clock of a design object when a related clock is not inferred by the related clock inference mechanism.

Data Types

Boolean

Default true in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Feedback

Description

This variable is used during the last stage of related clock inference to decide whether or not the design objects that do not have an inferred related clock will be set as a related clock.

You can use the *set_switching_activity* command with the *-clock "*"* argument to specify that the tool will automatically infer the related clocks for any specified objects. If the related clock inference mechanism did not infer a related clock for a number of such objects, then when the value of the *power_rclock_unrelated_use_fastest* is *true*, the tool will set the fastest design clock as the objects' related clock. When the variable is set to *false*, such objects will not have a related clock.

For more information on the mechanism used to infer related clock information, see the *propagate_switching_activity* command man page.

See Also

- propagate_switching_activity
- power_rclock_inputs_use_clocks_fanout
- power_rclock_use_asynch_inputs

---

power_rclock_use_asynch_inputs

Specifies whether the inferred related clock on an asynchronous pin of a flip-flop is used to determine the inferred related clock on the cell's outputs.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable is used during related clock inference to decide whether the inferred related clock on a flip-flop cell output considers the inferred related clocks on the cell's asynchronous inputs.

When this variable is set to *false*, the inferred related clock on a flip-flop cell output is the inferred related clock on the cell's clock pin. When this variable is set to *true*, the inferred related clock on a flip-flop cell output is the fastest inferred clock on the cell's clock pin and asynchronous input pins.

Feedback

For more information on the mechanism used to infer related clock information, see the *propagate_switching_activity* command man page.

See Also

- propagate_switching_activity
- power_rclock_inputs_use_clocks_fanout
- power_rclock_use_asynch_inputs

power_remove_redundant_clock_gates

Specifies to the *compile -incremental* and *physopt -incremental commands to remove redundant Synopsys clock gating cells.*

Data Types

Boolean

Default true in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable specifies whether to remove redundant Synopsys clock gating cells during incremental compile. A clock gate is considered redundant when it is always enabled. This is the case when the enable net is tied to logic one. When this variable is set to *true* (the default value), the redundant clock-gating cells are removed during incremental compile. To disable the automatic removal of redundant clock gating cells, set the value to *false* before issuing the *compile* or *physopt* commands.

To determine the current value of this variable, use the *printvar power_remove_redundant_clock_gates* command. For a list of power variables and their current values, use *print_variable_group power*.

See Also

- compile

power_report_separate_switching_power

Forces the tool to report wire switching power and pin switching power in two separate columns for the group report from report_power.

Feedback

Data Types

Boolean

Default false

Description

The *power_report_separate_switching_power* variable forces the tool to report wire switching power and pin switching power in two separate columns for the group report from report_power. By default, only one column is shown for switching power, which includes wire switching power and pin switching power.

The *power_report_separate_switching_power* variable is off (false) by default.

See Also

- report_power

---

power_same_switching_activity_on_connected_objects

Forces the tool to use the last user-annotated switching activity data on all connected tool objects.

Data Types

Boolean

Default false

Description

The *power_same_switching_activity_on_connected_objects* variable forces all switching activity data of all connected tool objects to use the latest user-annotated switching activity data. The user-annotated switching activity information can come from the *read_saif* command or the *set_switching_activity* command. Power reporting commands may produce different power results, since the switching activity data may be changed.

The *power_same_switching_activity_on_connected_objects* variable is off (false) by default.

See Also

- read_saif
- set_switching_activity

Feedback

power_scale_internal_arc

Enables scaling of state probabilities of internal power arcs.

Data Types

Boolean

Default false

Description

By default this variable is set to *false*, which disables scaling. If you set the Tcl variable to *true*, then the sum of the probabilities of the internal arc matches with the toggle rate of the pin even if there is no default arc for the pin in the library.

To find the current value of this variable, use the following command:

```
get_app_var power_scale_internal_arc
```

power_sdpd_message_tolerance

Specifies the tolerance value for issuing warning and information messages during fixing of user-annotated SDPD switching activity.

Data Types

float

Default 1e-05 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable specifies a tolerance value that is used when messages are reported during the SDPD fixing step. The SDPD fixing step is enabled by the *power_fix_sdpd_annotation* variable. See the *power_fix_sdpd_annotation* command man page for more information on this step.

Messages reported by the SDPD fixing step need to specify the tolerance criteria specified by this variable. For example, SDPD fixing scales state-dependent static probabilities so that they add up to 1.0.

A message is reported by the SDPD fixing step if the difference between the sum of the state-dependent static probabilities and 1.0 is not within the tolerance value. A

Feedback

tolerance value of 0.0 makes the SDPD fixing step report a message for every check and modification to the SDPD annotation; however, this will most likely report non-issues due to floating point errors. A high value of the *power_sdpd_message_tolerance* value filters out all but the most significant messages. Note that the verbosity of the SDPD fixing step is determined by the *power_fix_sdpd_annotation_verbose* variable. When *power_fix_sdpd_annotation_verbose* is set to *false*, most of the messages reported by the SDPD fixing step are replaced by a single message indicating that user-annotated SDPD switching activity is being corrected.

See Also

- power_fix_sdpd_annotation
- power_fix_sdpd_annotation_verbose

---

power_write_saif_keep_ungrouped_name

Will keep the ungrouped names during write_saif for those instances which have been ungrouped.

Data Types

Boolean

Default false

Description

By default after an ungrouping operation the original structure is used in the output file from write_saif. When setting this variable to true, the tool will keep the ungrouped instance name, aligning with the resulting Verilog hierarchies.

Examples

Use the following flow to write out a SAIF file:

```
prompt> ungroup -flatten -all
prompt> compile_ultra -spg
prompt> set_app_var power_write_saif_keep_ungrouped_name false
prompt> rite_saif convention1.saif
prompt> h cat ./convention1.saif
(INSTANCE top
   (INSTANCE MID
      (INSTANCE BOT

prompt> set_app_var power_write_saif_keep_ungrouped_name true
prompt> rite_saif convention2.saif
prompt> h cat ./convention2.saif
(INSTANCE top
   (INSTANCE MID\/INSTANCE BOT
```

Feedback

preroute_opt_verbose

Controls how verbose messages are displayed during DRC fixing, hold fixing, multiple-port-net fixing and tie-off optimization in the preroute stage. The debug can be used for debug purposes.

Data Types

Positive integer

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

0 in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

This variable controls how verbose messages are displayed during DRC fixing, hold fixing, multiple-port-net fixing, tie-off and setup optimization in preroute stage. The messages can be used for debug purposes.

The default is 0.

When the variable is set to bitwise AND 2 (bit 2), which equals 1, the verbose messages are displayed during DRC fixing. For example, *set preroute_opt_verbose 2*.

When the variable is set to bitwise AND 4 (bit 3), which equals 1, the verbose messages are displayed during hold fixing. For example, *set preroute_opt_verbose 4*.

When the variable is set to bitwise AND 8 (bit 4), which equals 1, the verbose message are displayed during tie-off optimization. For example, *set preroute_opt_verbose 8*.

When the variable is set to bitwise AND 16 (bit 5), which equals 1, the verbose message are displayed during multiple-port-net fixing. For example, *set preroute_opt_verbose 16*.

When the variable is set to bitwise AND 32 (bit 6), which equals 1, the verbose message are displayed during setup fixing. For example, *set preroute_opt_verbose 32*.

When the variable is set to bitwise AND 128 (bit 8), which equals 1, the verbose message will be saved to file propt_verbose.log. Bit 8 only works with DRC and setup fixing. For example, set preroute_opt_verbose 0xa0 (output setup verbose message to file) set preroute_opt_verbose 0x82 (output DRC verbose message to file)

You can output verbose message to a file. The file name is by default set as propt_verbose.log and can not be changed.

You can use hexadecimal format as input , for example: set preroute_opt_verbose 0x2 (DRC verbose) set preroute_opt_verbose 0x4 (hold verbose) set preroute_opt_verbose

Feedback

0x8 (tie-off verbose) set preroute_opt_verbose 0x10 (mpn verbose) set preroute_opt_verbose 0x20 (setup verbose)

A common error from usage is: set preroute_opt_verbose 8002

Above setting is not valid value infact and the result is unknown.

You can also set the variable to show messages in multiple areas. For example, set preroute_opt_verbose 24 (both tie-off and mpn verbose)

---

preserve_collections_in_compile

Attempt to preserve collections during compile

Data Types

Boolean

Default true in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

false in DC Explorer

Description

By default this value is set to true. During *compile* optimizations and changes may invalidate existing collections (due to memory reference changes). If this variable is set to true, all the references will be kept updated as best as possible, though it adds a runtime overhead. Turning this off, reduces this overhead at the cost of existing collections being invalid after *compile*.

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

```
prompt> printvar preserve_collections_in_compile
```

See Also

- compile

---

psynopt_density_limit

Sets the density limit for local region optimization. This limit prevents preroute optimization being performed on high density regions.

Data Types

float

Feedback

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

-1 in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Group

psynopt_variables

Description

This variable sets the density limit for checking a local region during preroute optimization stage.

When the tool tries to optimize a local region, it checks the region's density. If the density is greater or equal to the limit, then the density check fails, and the optimization process is NOT be performed. That means, the optimization tool does NOT perform buffer insertion or cell sizing in this high density region.

The variable affects the following optimization processes: power recovery, DRC fixing, timing fixing, hold fixing, and area recovery.

The variable affects the following optimization commands: psynopt, place_opt, clock_opt, preroute_focal_opt, place_opt_feasibility, and clock_opt_feasibility.

The default is -1. You should set the variable to a number less than 1.0. If you set the variable to a number larger than 1.0, the optimization might result in high local density.

---

psynopt_tns_high_effort

Enables high-effort optimization to improve total negative slack in the design when you run a preroute command, such as *psynopt* or *place_opt*.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

When you set the *psynopt_tns_high_effort* variable to *true*, preroute commands, such as *place_opt*, *clock_opt*, and *psynopt* use high-effort optimization strategies to improve total negative slack (TNS). This variable is not enabled by default because it increases runtime. Set the variable to *true if further TNS optimization is required.*

This variable also improves TNS in the *compile_ultra* command, but the impact is minimal. To enable high-effort TNS optimization in *compile_ultra*, use the *compile_timing_high_effort_tns variable.*

Examples

The following example enables the *place_opt* command to use high-effort optimization to improve total negative slack:

```
prompt> set_app_var psynopt_tns_high_effort true
prompt> psynopt
```

q

query_objects_format

Data Types

string

Default Legacy

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

Feedback

r

rc_degrade_min_slew_when_rd_less_than_rnet

Enables or disables the use of slew degradation in minimum analysis mode during the RCCALC-009 condition.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable enables the use of slew degradation during the RCCALC-009 condition. When set to *false*, (the default), slew degradation through RC networks is not used in minimum analysis mode during the RCCALC-009 condition.

The "RCCALC-009 condition" means a condition in which timing analysis checks the library-derived drive resistance, and if it is less than the dynamic RC network impedance to ground by an amount equal to or greater than the value of a particular drive-strength threshold, timing analysis adjusts the drive resistance using an empirical formula to improve accuracy, and issues the RCCALC-009 message. If this improved accuracy is not sufficient, timing analysis provides extra pessimism by not using slew degradation in minimum analysis mode; however, unnecessary minimum delay violations could occur as a side effect. You can keep slew degradation on in minimum analysis mode after you have qualified the RCCALC-009 methodology for your accuracy requirements, by setting this variable to *true*.

To determine the current value of this variable, use the *printvar rc_degrade_min_slew_when_rd_less_than_rnet* or *echo $rc_degrade_min_slew_when_rd_less_than_rnet* command.

rc_driver_model_mode

Specifies the driver model type to use for RC delay calculation.

Data Types

string

Default advanced

Feedback

Description

This variable specifies whether the timing engine uses a basic or advanced driver model for RC delay calculation. The *basic* model is derived from a simpler delay and slew library method, whereas the *advanced* model is derived from an advanced driver model that is part of the Synopsys Composite Current Source (CCS) model.

When the variable is set to *basic*, RC delay calculation always uses driver models derived from the conventional delay and slew model present in the design library. When it is set to *advanced* (the default), RC delay calculation uses the advanced driver model, if data for the model is present. In that case, when you use the *report_delay_calculation* command to report the cell arc, it displays the message "Advanced driver-modeling used".

You do not need to set the *rc_driver_model_mode* variable to *advanced* to enable the CCS driver model. The CCS model is used automatically if the CCS libraries exist. However, you can set the *rc_driver_model_mode* variable to *basic* to disable the advanced CCS driver model.

When the *rc_driver_model_mode* variable is set to *basic* and the *rc_receiver_model_mode* variable is set to *advanced*, the timing engine uses the advanced voltage-dependent capacitance models to derive an equivalent single capacitance model that depends only on the rise, fall, minimum, or maximum arc condition. These equivalent capacitance values are used in the analysis instead of the pin capacitance values from the library. For details, see the *rc_receiver_model_mode* variable man page.

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

See Also

- report_delay_calculation
- rc_receiver_model_mode

---

rc_input_threshold_pct_fall

Specifies the threshold voltage that defines the startpoint of the falling cell or net delay calculation.

Data Types

float

Default 50 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Feedback

Description

This variable specifies the threshold voltage that defines the startpoint of the falling cell or net delay calculation. The value is a percent of the voltage source. Allowed values are 0.0 through 100.0 inclusive.

This variable is one of 8 variables, listed in Table 1, that affect delay and transition time computations for detailed RC networks. These variables interpret the cell delays and transition times from the Synopsys library. The values specified by these variables are overridden by trip-point values specified in the library, so it should not be necessary to set the variables. The values specified are only applied to libraries that do not contain trip-point specifications.

Table 1

Variable Name	Default
rc_slew_lower_threshold_pct_rise	20.0
rc_slew_lower_threshold_pct_fall	20.0
rc_slew_upper_threshold_pct_rise	80.0
rc_slew_upper_threshold_pct_fall	80.0
rc_input_threshold_pct_rise	50.0
rc_input_threshold_pct_fall	50.0
rc_output_threshold_pct_rise	50.0
rc_output_threshold_pct_fall	50.0

The default values specify that a cell delay is defined from 50 percent of the voltage value for the input transition to 50 percent of the voltage value for the output transition. The default values also specify that a transition time, or slew, is defined from 20 percent to 80 percent of the voltage.

To determine the current value of this variable, use the *printvar rc_input_threshold_pct_fall* command. For a list of all timing variables and their current values, use *print_variable_group timing*.

Examples

The following example specifies that cell delays from the Synopsys library are computed from 50 percent of the input transition to 55 percent of the output transition. In addition, the example specifies that transition times in the Synopsys library represent the delay from 10 percent to 90 percent of the voltage source.

```
prompt> set rc_slew_lower_threshold_pct_fall 10
prompt> set rc_slew_lower_threshold_pct_rise 10
prompt> set rc_slew_upper_threshold_pct_fall 90
prompt> set rc_slew_upper_threshold_pct_rise 90
prompt> set rc_input_threshold_pct_rise      50
prompt> set rc_input_threshold_pct_fall      50
prompt> set rc_output_threshold_pct_rise     55
prompt> set rc_output_threshold_pct_fall     55
```

Feedback

See Also

- rc_input_threshold_pct_fall
- rc_input_threshold_pct_rise
- rc_output_threshold_pct_fall
- rc_output_threshold_pct_rise
- rc_slew_lower_threshold_pct_fall
- rc_slew_lower_threshold_pct_rise
- rc_slew_upper_threshold_pct_fall
- rc_slew_upper_threshold_pct_rise

---

rc_input_threshold_pct_rise

Specifies the threshold voltage that defines the startpoint of the rising cell or net delay calculation.

Data Types

float

Default 50 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable specifies the threshold voltage that defines the startpoint of the rising cell or net delay calculation. The value is a percent of the voltage source. Allowed values are 0.0 through 100.0 inclusive.

This variable is one of 8 variables, listed in Table 1, that affect delay and transition time computations for detailed RC networks. These variables interpret the cell delays and transition times from the Synopsys library. The values specified by these variables are overridden by trip-point values specified in the library, so normally it should not be necessary to set the variables. The values specified are only applied to libraries that do not contain trip-point specifications.

Table 1

Variable Name	Default
rc_slew_lower_threshold_pct_rise	20.0

Feedback

```
rc_slew_lower_threshold_pct_fall        20.0
rc_slew_upper_threshold_pct_rise        80.0
rc_slew_upper_threshold_pct_fall        80.0
rc_input_threshold_pct_rise             50.0
rc_input_threshold_pct_fall             50.0
rc_output_threshold_pct_rise            50.0
rc_output_threshold_pct_fall            50.0
```

The default values specify that a cell delay is defined from 50 percent of the voltage value for the input transition to 50 percent of the voltage value for the output transition. The default values also specify that a transition time, or slew, is defined from 20 percent to 80 percent of the voltage.

To determine the current value of this variable, use the *printvar rc_input_threshold_pct_rise* command. For a list of all timing variables and their current values, use *print_variable_group timing*.

Examples

The following example specifies that cell delays from the Synopsys library are computed from 50 percent of the input transition to 55 percent of the output transition. In addition, the example specifies that transition times in the Synopsys library represent the delay from 10 percent to 90 percent of the voltage source.

```
prompt> set rc_slew_lower_threshold_pct_fall 10
prompt> set rc_slew_lower_threshold_pct_rise 10
prompt> set rc_slew_upper_threshold_pct_fall 90
prompt> set rc_slew_upper_threshold_pct_rise 90
prompt> set rc_input_threshold_pct_rise      50
prompt> set rc_input_threshold_pct_fall      50
prompt> set rc_output_threshold_pct_rise     55
prompt> set rc_output_threshold_pct_fall     55
```

See Also

- rc_input_threshold_pct_fall
- rc_input_threshold_pct_rise
- rc_output_threshold_pct_fall
- rc_output_threshold_pct_rise
- rc_slew_lower_threshold_pct_fall
- rc_slew_lower_threshold_pct_rise
- rc_slew_upper_threshold_pct_fall
- rc_slew_upper_threshold_pct_rise

Feedback

rc_noise_model_mode

Enables the use of CCS noise, if available in the design library, when set to *advanced*.

Data Types

string

Default basic in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Group

signal integrity

Description

This variable enables CCS noise in static noise analysis and optimization, when set to *advanced*.

When set to *basic* (the default), CCS noise information is not used even if it exists in the library. However, if the library has NLDM noise information, it is still used.

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

See Also

- rc_driver_model_mode
- rc_receiver_model_mode

rc_output_threshold_pct_fall

Specifies the threshold voltage that defines the startpoint of the falling cell or net delay calculation.

Data Types

float

Default 50 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Feedback

Description

This variable specifies the threshold voltage that defines the startpoint of the falling cell or net delay calculation. The value is a percent of the voltage source. Allowed values are 0.0 through 100.0 inclusive.

This variable is one of 8 variables, listed in Table 1, that affect delay and transition time computations for detailed RC networks. These variables interpret the cell delays and transition times from the Synopsys library. The values specified by these variables are overridden by trip-point values specified in the library, so it should not be necessary to set the variables. The values specified are only applied to libraries that do not contain trip-point specifications.

Table 1

Variable Name	Default
rc_slew_lower_threshold_pct_rise	20.0
rc_slew_lower_threshold_pct_fall	20.0
rc_slew_upper_threshold_pct_rise	80.0
rc_slew_upper_threshold_pct_fall	80.0
rc_input_threshold_pct_rise	50.0
rc_input_threshold_pct_fall	50.0
rc_output_threshold_pct_rise	50.0
rc_output_threshold_pct_fall	50.0

The default values specify that a cell delay is defined from 50 percent of the voltage value for the input transition to 50 percent of the voltage value for the output transition. The default values also specify that a transition time, or slew, is defined from 20 percent to 80 percent of the voltage.

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

For a list of all timing variables and their current values, type *print_variable_group timing*.

Examples

The following example specifies that cell delays from the Synopsys library are computed from 50 percent of the input transition to 55 percent of the output transition. In addition, the example specifies that transition times in the Synopsys library represent the delay from 10 percent to 90 percent of the voltage source.

```
prompt> set rc_slew_lower_threshold_pct_fall 10
prompt> set rc_slew_lower_threshold_pct_rise 10
prompt> set rc_slew_upper_threshold_pct_fall 90
prompt> set rc_slew_upper_threshold_pct_rise 90
prompt> set rc_input_threshold_pct_rise      50
prompt> set rc_input_threshold_pct_fall      50
```

Feedback

```
prompt> set rc_output_threshold_pct_rise     55
prompt> set rc_output_threshold_pct_fall     55
```

See Also

- rc_input_threshold_pct_fall
- rc_input_threshold_pct_rise
- rc_output_threshold_pct_fall
- rc_output_threshold_pct_rise
- rc_slew_lower_threshold_pct_fall
- rc_slew_lower_threshold_pct_rise
- rc_slew_upper_threshold_pct_fall
- rc_slew_upper_threshold_pct_rise

---

rc_output_threshold_pct_rise

Specifies the threshold voltage that defines the startpoint of the rising cell or net delay calculation.

Data Types

float

Default 50 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable specifies the threshold voltage that defines the startpoint of the rising cell or net delay calculation. The value is a percent of the voltage source. Allowed values are 0.0 through 100.0 inclusive.

This variable is one of 8 variables, listed in Table 1, that affect delay and transition time computations for detailed RC networks. These variables interpret the cell delays and transition times from the Synopsys library. The values specified by these variables are overridden by trip-point values specified in the library, so it should not be necessary to set the variables. The values specified are only applied to libraries that do not contain trip-point specifications.

```
Table 1

Variable Name                                  Default
-------------------------------------------------------
rc_slew_lower_threshold_pct_rise               20.0
rc_slew_lower_threshold_pct_fall               20.0
rc_slew_upper_threshold_pct_rise               80.0
rc_slew_upper_threshold_pct_fall               80.0
rc_input_threshold_pct_rise                    50.0
rc_input_threshold_pct_fall                    50.0
rc_output_threshold_pct_rise                   50.0
rc_output_threshold_pct_fall                   50.0
```

The default values specify that a cell delay is defined from 50 percent of the voltage value for the input transition to 50 percent of the voltage value for the output transition. The default values also specify that a transition time, or slew, is defined from 20 percent to 80 percent of the voltage.

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

For a list of all timing variables and their current values, use *print_variable_group timing*.

Examples

The following example specifies that cell delays from the Synopsys library are computed from 50 percent of the input transition to 55 percent of the output transition. In addition, the example specifies that transition times in the Synopsys library represent the delay from 10 percent to 90 percent of the voltage source.

```
prompt> set rc_slew_lower_threshold_pct_fall 10
prompt> set rc_slew_lower_threshold_pct_rise 10
prompt> set rc_slew_upper_threshold_pct_fall 90
prompt> set rc_slew_upper_threshold_pct_rise 90
prompt> set rc_input_threshold_pct_rise      50
prompt> set rc_input_threshold_pct_fall      50
prompt> set rc_output_threshold_pct_rise     55
prompt> set rc_output_threshold_pct_fall     55
```

See Also

- rc_input_threshold_pct_fall
- rc_input_threshold_pct_rise
- rc_output_threshold_pct_fall
- rc_output_threshold_pct_rise
- rc_slew_lower_threshold_pct_fall

Feedback

- rc_slew_lower_threshold_pct_rise
- rc_slew_upper_threshold_pct_fall
- rc_slew_upper_threshold_pct_rise

---

rc_receiver_model_mode

Specifies the receiver model type to use for RC delay calculation.

Data Types

string

Default advanced

Description

This variable specifies whether the timing engine uses a basic or advanced receiver model for RC delay calculation. The *basic* model is a single capacitance model that depends only on the rise, fall, minimum, or maximum arc condition. The *advanced* model is a voltage-dependent capacitance model that also depends on input slew and output capacitance.

One advantage of the advanced model is improved accuracy for both delay and slew calculation. Another advantage is that it considers nonlinear effects such as the Miller effect. The advanced receiver model is part of the Synopsys Composite Current Source (CCS) model.

When this variable is set to *advanced* (the default), RC delay calculation uses the advanced receiver model if data for that model is present and if the network driver uses the advanced driver model. In that case, a report generated by the *report_delay_calculation* command for a network arc displays the message "Advanced receiver-modeling used".

You do not need to set the *rc_receiver_model_mode* variable to *advanced* to enable the CCS receiver model. The CCS model is used automatically if the CCS libraries exist. However, you can set the *rc_receiver_model_mode* variable to *basic* to disable the advanced CCS receiver model.

When the *rc_receiver_model_mode* variable is set to *advanced*, and the network is not driven by the advanced driver model (such as when the *rc_driver_model_mode* variable is set to basic or a lumped load is used), the timing engine uses the advanced voltage-dependent capacitance models to derive an equivalent single capacitance model that depends only on the rise, fall, minimum, or maximum arc condition. These equivalent capacitance values are used in analysis instead of the pin capacitance values from the library. The *report_delay_calculation* command used on a network arc does not show the "Advanced receiver-modeling used" message for these calculations because only an equivalent single capacitance is used.

When the *rc_receiver_model_mode* variable is set to *basic*, RC delay calculation uses the pin capacitance values specified in the design libraries.

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

See Also

- report_delay_calculation
- rc_driver_model_mode

---

rc_slew_derate_from_library

Specifies the derating needed for the transition times in the Synopsys library to match the transition times between the characterization trip points.

Data Types

float

Default 1 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable specifies a floating-point number between 0.0 and 1.0 that indicates the derating needed for the transition times in the Synopsys library to match the transition times between the characterization trip points. The default is 1.0, which means that the transition times in the Synopsys library are used without change.

The value this variable specifies is overridden by any library-specified slew-derating values, so it should not be necessary to set the variable. The value specified applies only to libraries that do not contain slew-derating specifications.

A slew-derating value of 1.0 should be used if the transition times specified in the library represent the exact transition times between the characterization trip points, which is usually the case. Use a slew-derating value of less than 1.0 for libraries where the transition times have been extrapolated to the rail voltages. For example, if the transition times are characterized as between 30 percent and 70 percent and then extrapolated to the rails, the slew-derating value is 0.4 = (70 - 30) / 100.

To determine the current value of this variable, use the *printvar rc_slew_derate_from_library* command. For a list of all timing variables and their current values, use *print_variable_group timing*.

Feedback

Examples

The following example specifies that cell delays from the Synopsys library have been computed from 50 percent of the input transition to 50 percent of the output transition. The example also specifies that transition times in the Synopsys library were computed by measuring the delay from 30 percent to 70 percent of the voltage source and then multiplying the measured transition times by 2.5 = (100-0)/(70-30) to extrapolate to 0-100 percent of the rail voltages.

```
prompt> set rc_slew_derate_from_library 0.4
prompt> set rc_slew_lower_threshold_pct_fall 30
prompt> set rc_slew_lower_threshold_pct_rise 30
prompt> set rc_slew_upper_threshold_pct_fall 70
prompt> set rc_slew_upper_threshold_pct_rise 70
prompt> set rc_input_threshold_pct_rise      50
prompt> set rc_input_threshold_pct_fall      50
prompt> set rc_output_threshold_pct_rise     55
prompt> set rc_output_threshold_pct_fall     55
```

See Also

- rc_input_threshold_pct_fall
- rc_input_threshold_pct_rise
- rc_output_threshold_pct_fall
- rc_output_threshold_pct_rise
- rc_slew_lower_threshold_pct_fall
- rc_slew_lower_threshold_pct_rise
- rc_slew_upper_threshold_pct_fall
- rc_slew_upper_threshold_pct_rise

---

rc_slew_lower_threshold_pct_fall

Specifies the threshold voltage that defines the endpoint of the falling slew calculation.

Data Types

float

Default 20 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Feedback

Description

This variable specifies the threshold voltage that defines the endpoint of the falling slew calculation. The value is a percent of the voltage source. Allowed values are 0.0 through 100.0 inclusive.

This variable is one of 8 variables, listed in Table 1, that affect delay and transition time computations for detailed RC networks. These variables interpret the cell delays and transition times from the Synopsys library. The values specified by these variables are overridden by trip-point values specified in the library, so it should not be necessary to set the variables. The values specified are only applied to libraries that do not contain trip-point specifications.

Table 1

Variable Name	Default
rc_slew_lower_threshold_pct_rise	20.0
rc_slew_lower_threshold_pct_fall	20.0
rc_slew_upper_threshold_pct_rise	80.0
rc_slew_upper_threshold_pct_fall	80.0
rc_input_threshold_pct_rise	50.0
rc_input_threshold_pct_fall	50.0
rc_output_threshold_pct_rise	50.0
rc_output_threshold_pct_fall	50.0

The default values specify that a cell delay is defined from 50 percent of the voltage value for the input transition to 50 percent of the voltage value for the output transition. The default values also specify that a transition time, or slew, is defined from 20 percent to 80 percent of the voltage.

To determine the current value of this variable, use the *printvar rc_slew_lower_threshold_pct_fall* command. For a list of all timing variables and their current values, use *print_variable_group timing*.

Examples

The following example specifies that cell delays from the Synopsys library are computed from 50 percent of the input transition to 55 percent of the output transition. In addition, the example specifies that transition times in the Synopsys library represent the delay from 10 percent to 90 percent of the voltage source.

```
prompt> set rc_slew_lower_threshold_pct_fall 10
prompt> set rc_slew_lower_threshold_pct_rise 10
prompt> set rc_slew_upper_threshold_pct_fall 90
prompt> set rc_slew_upper_threshold_pct_rise 90
prompt> set rc_input_threshold_pct_rise      50
prompt> set rc_input_threshold_pct_fall      50
prompt> set rc_output_threshold_pct_rise     55
prompt> set rc_output_threshold_pct_fall     55
```

Feedback

See Also

- rc_input_threshold_pct_fall
- rc_input_threshold_pct_rise
- rc_output_threshold_pct_fall
- rc_output_threshold_pct_rise
- rc_slew_lower_threshold_pct_fall
- rc_slew_lower_threshold_pct_rise
- rc_slew_upper_threshold_pct_fall
- rc_slew_upper_threshold_pct_rise

---

rc_slew_lower_threshold_pct_rise

Specifies the threshold voltage that defines the startpoint of the rising slew calculation.

Data Types

float

Default 20 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable specifies the threshold voltage that defines the startpoint of the rising slew calculation. The value is a percent of the voltage source. Allowed values are 0.0 through 100.0 inclusive.

This variable is one of 8 variables, listed in Table 1, that affect delay and transition time computations for detailed RC networks. These variables interpret the cell delays and transition times from the Synopsys library. The values specified by these variables are overridden by trip-point values specified in the library, so it should not be necessary to set the variables. The values specified are only applied to libraries that do not contain trip-point specifications.

Table 1

Variable Name	Default
rc_slew_lower_threshold_pct_rise	20.0
rc_slew_lower_threshold_pct_fall	20.0

```
rc_slew_upper_threshold_pct_rise        80.0
rc_slew_upper_threshold_pct_fall        80.0
rc_input_threshold_pct_rise             50.0
rc_input_threshold_pct_fall             50.0
rc_output_threshold_pct_rise            50.0
rc_output_threshold_pct_fall            50.0
```

The default values specify that a cell delay is defined from 50 percent of the voltage value for the input transition to 50 percent of the voltage value for the output transition. The default values also specify that a transition time, or slew, is defined from 20 percent to 80 percent of the voltage.

To determine the current value of this variable, use the *printvar rc_slew_lower_threshold_pct_rise* command. For a list of all timing variables and their current values, use *print_variable_group timing*.

Examples

The following example specifies that cell delays from the Synopsys library are computed from 50 percent of the input transition to 55 percent of the output transition. In addition, the example specifies that transition times in the Synopsys library represent the delay from 10 percent to 90 percent of the voltage source.

```
prompt> set rc_slew_lower_threshold_pct_fall 10
prompt> set rc_slew_lower_threshold_pct_rise 10
prompt> set rc_slew_upper_threshold_pct_fall 90
prompt> set rc_slew_upper_threshold_pct_rise 90
prompt> set rc_input_threshold_pct_rise      50
prompt> set rc_input_threshold_pct_fall      50
prompt> set rc_output_threshold_pct_rise     55
prompt> set rc_output_threshold_pct_fall     55
```

See Also

- rc_input_threshold_pct_fall
- rc_input_threshold_pct_rise
- rc_output_threshold_pct_fall
- rc_output_threshold_pct_rise
- rc_slew_lower_threshold_pct_fall
- rc_slew_lower_threshold_pct_rise
- rc_slew_upper_threshold_pct_fall
- rc_slew_upper_threshold_pct_rise

Feedback

rc_slew_upper_threshold_pct_fall

Specifies the threshold voltage that defines the startpoint of the falling slew calculation.

Data Types

float

Default 80 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable specifies the threshold voltage that defines the startpoint of the falling slew calculation. The value is a percent of the voltage source. Allowed values are 0.0 through 100.0 inclusive.

This variable is one of 8 variables, listed in Table 1, that affect delay and transition time computations for detailed RC networks. These variables interpret the cell delays and transition times from the Synopsys library. The values specified by these variables are overridden by trip-point values specified in the library, so it should not be necessary to set the variables. The values specified are only applied to libraries that do not contain trip-point specifications.

Table 1

Variable Name	Default
rc_slew_lower_threshold_pct_rise	20.0
rc_slew_lower_threshold_pct_fall	20.0
rc_slew_upper_threshold_pct_rise	80.0
rc_slew_upper_threshold_pct_fall	80.0
rc_input_threshold_pct_rise	50.0
rc_input_threshold_pct_fall	50.0
rc_output_threshold_pct_rise	50.0
rc_output_threshold_pct_fall	50.0

The default values specify that a cell delay is defined from 50 percent of the voltage value for the input transition to 50 percent of the voltage value for the output transition. The default values also specify that a transition time, or slew, is defined from 20 percent to 80 percent of the voltage.

To determine the current value of this variable, use the *printvar rc_slew_upper_threshold_pct_fall* command. For a list of all timing variables and their current values, use *print_variable_group timing*.

Feedback

Examples

The following example specifies that cell delays from the Synopsys library are computed from 50 percent of the input transition to 55 percent of the output transition. In addition, the example specifies that transition times in the Synopsys library represent the delay from 10 percent to 90 percent of the voltage source.

```
prompt> set rc_slew_lower_threshold_pct_fall 10
prompt> set rc_slew_lower_threshold_pct_rise 10
prompt> set rc_slew_upper_threshold_pct_fall 90
prompt> set rc_slew_upper_threshold_pct_rise 90
prompt> set rc_input_threshold_pct_rise      50
prompt> set rc_input_threshold_pct_fall      50
prompt> set rc_output_threshold_pct_rise     55
prompt> set rc_output_threshold_pct_fall     55
```

See Also

- rc_input_threshold_pct_fall
- rc_input_threshold_pct_rise
- rc_output_threshold_pct_fall
- rc_output_threshold_pct_rise
- rc_slew_lower_threshold_pct_fall
- rc_slew_lower_threshold_pct_rise
- rc_slew_upper_threshold_pct_fall
- rc_slew_upper_threshold_pct_rise

---

rc_slew_upper_threshold_pct_rise

Specifies the threshold voltage that defines the endpoint of the rising slew calculation.

Data Types

float

Default 80 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Feedback

Description

This variable specifies the threshold voltage that defines the endpoint of the rising slew calculation. The value is a percent of the voltage source. Allowed values are 0.0 through 100.0 inclusive.

This variable is one of 8 variables, listed in Table 1, that affect delay and transition time computations for detailed RC networks. These variables interpret the cell delays and transition times from the Synopsys library. The values specified by these variables are overridden by trip-point values specified in the library, so it should not be necessary to set the variables. The values specified are only applied to libraries that do not contain trip-point specifications.

Table 1

Variable Name	Default
rc_slew_lower_threshold_pct_rise	20.0
rc_slew_lower_threshold_pct_fall	20.0
rc_slew_upper_threshold_pct_rise	80.0
rc_slew_upper_threshold_pct_fall	80.0
rc_input_threshold_pct_rise	50.0
rc_input_threshold_pct_fall	50.0
rc_output_threshold_pct_rise	50.0
rc_output_threshold_pct_fall	50.0

The default values specify that a cell delay is defined from 50 percent of the voltage value for the input transition to 50 percent of the voltage value for the output transition. The default values also specify that a transition time, or slew, is defined from 20 percent to 80 percent of the voltage.

To determine the current value of this variable, use the *printvar rc_slew_upper_threshold_pct_rise* command. For a list of all timing variables and their current values, use *print_variable_group timing*.

Examples

The following example specifies that cell delays from the Synopsys library are computed from 50 percent of the input transition to 55 percent of the output transition. In addition, the example specifies that transition times in the Synopsys library represent the delay from 10 percent to 90 percent of the voltage source.

```
prompt> set rc_slew_lower_threshold_pct_fall 10
prompt> set rc_slew_lower_threshold_pct_rise 10
prompt> set rc_slew_upper_threshold_pct_fall 90
prompt> set rc_slew_upper_threshold_pct_rise 90
prompt> set rc_input_threshold_pct_rise      50
prompt> set rc_input_threshold_pct_fall      50
prompt> set rc_output_threshold_pct_rise     55
prompt> set rc_output_threshold_pct_fall     55
```

Feedback

See Also

- rc_input_threshold_pct_fall
- rc_input_threshold_pct_rise
- rc_output_threshold_pct_fall
- rc_output_threshold_pct_rise
- rc_slew_lower_threshold_pct_fall
- rc_slew_lower_threshold_pct_rise
- rc_slew_upper_threshold_pct_fall
- rc_slew_upper_threshold_pct_rise

---

read_db_lib_warnings

Indicates that warnings are to be printed while a technology .db library is being read in with the *read* command. When false (the default), no warnings are given.

Data Types

Boolean

Default false in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Indicates that warnings are to be printed while a technology .db library is being read in with the *read* command. When false (the default), no warnings are given.

To determine the current value of this variable, type *printvar read_db_lib_warnings*. For a list of all *io* variables and their current values, type *print_variable_group io*.

See Also

- read

---

read_translate_msff

Automatically translates master-slave flip-flops (specified with the clocked_on_also syntax) to master-slave latches.

Feedback

Data Types

Boolean

Default true in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable automatically translates master-slave flip-flops (specified with the clocked_on_also syntax) to master-slave latches, when set to *true* (the default). When set to *false*, both master and slave remain flip-flops.

This variable is used when running the *read_file* command, while a technology .db library is being read in by the shell, and when running the *read_lib* command while a technology library is being read in by Library Compiler. The technology .db library is affected only if the program reports that the .db library is being updated and asks you to save the results. Library Compiler always follows this variable during processing.

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

See Also

- read_file
- read_lib

---

register_duplicate

Specifies that the *compile* command is to invoke register duplication to reduce the number of fanouts for each register.

Data Types

Boolean

Default false

Description

The *register_duplicate* variable when set to *true*(default is false), duplicates the high fanout registers to reduce the number of fanouts for each register. The max fanout that each register should drive is specified by setting set_max_fanout. The register replication occurs provided that you specified max fanout constraint using set_max_fanout.

Feedback

Fanouts for registers that exceed a certain limit can cause an adverse affect on the circuit performance. The fanout is by default controlled only by buffer insertion. Some FPGA architectures lack buffers in the routing architecture that might cause the fixed fanout limit to exceed and cause an error in the back-end tool. To address this problem, the *register_duplicate* variable provides a method to duplicate registers in an optimized manner.

Examples

The following is an example of the use of *register_duplicate*:

```
prompt> set_max_fanout 25 top

prompt> set_app_var register_duplicate true

prompt> set_attribute xfpga_virtex2-5 \\
        -type float default_fanout_load 100
```

The *compile* command indicates the register duplication status as ":-". After the report is generated, the following message appears:

```
Information: Duplicating register r1_reg with fanout load of 50.00
 (REGDUP-3)

Optimization Complete
```

See Also

- compile
- set_max_fanout

---

register_replication_naming_style

Specifies the style to use in naming the replicated register with the *set_register_replication* command.

Data Types

string

Default %s_rep%d

Description

This variable specifies the naming style of the replicated register created by the *set_register_replication* command. It is a string that has exactly one %s and one %d. The %s is replaced by the base name of the register. The %d is the i-th copy of the replicated register.

Feedback

For example, if the variable is %s_rep_%d, then the first copy of the u0_reg register is u0_reg_rep_1. The fifth copy of the u6 register is u6_rep_5.

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

See Also

- set_register_replication

---

remove_constant_register

is an attribute which specifies whether constant register is set to be optimized or preserved which is set by set_constant_register_removal.

Data Types

Boolean

Default false

Description

This attribute being set by set_constant_register_removal command when set to false preserves the constant register, true allows optimization.

---

remove_unloaded_register

is an attribute which specifies whether unloaded register is set to be optimized or preserved which is set by set_unloaded_register_removal.

Data Types

Boolean

Default false

Description

This attribute being set by set_unloaded_register_removal command when set to false preserves the unloaded register, true allows optimization.

---

report_capacitance_use_ccs_receiver_model

Specifies whether the basic or advanced receiver model is used to report receiver pin capacitance.

Feedback

Data Types

Boolean

Default true

Description

When set this variable to be true, the advanced CCS receiver model is used to report receiver pin capacitance by the *report_net*, *report_constraint*, *report_delay_calculation*, and *report_timing* commands. When it is set to false, the basic library-derived lumped pin capacitance is used. The variable setting only affects pin capacitance reporting, not delay calculation.

See Also

- report_timing
- report_net
- report_constraint
- report_delay_calculation

---

report_default_significant_digits

Sets the default number of significant digits for many reports.

Data Types

integer

Default -1

Description

The *report_default_significant_digits* variable sets the default number of significant digits for many reports. Allowed values are 0-13; the default is -1. A value of -1 indicates that a command-specific default precision value is used for reporting. Some report commands, such as *report_timing* and *report_cell*, have a *-significant_digits* option, which overrides the value of this variable.

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

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

Once set, the value of the variable is used by the subsequent reporting commands if a command-specific *-significant_digits* option is not used. The value of the variable can be

reset by setting the value of the variable to -1. This causes the command-specific default to be used as the precision for reporting if the *-significant_digits* command option is not used.

For example, assume the command-specific default precision of the *report_cell* command is 6. If the *report_default_significant_digits* variable is set to 5 and the *report_cell* command is run thereafter, a precision of 5 is used in the *report_cell* report. If the *report_cell -significant_digits 3* command is run after the above variable setting, a precision of 3 is used for the *report_cell* report. To view the *report_cell* report with the command default precision of 6, reset the value of the *report_default_significant_digits* variable by setting it to -1.

If an invalid value is set (not in the range 0-13 and not -1), an error is issued and the previous valid value is restored. The following example illustrates the usage of the variable:

```
prompt> printvar report_default_significant_digits
report_default_significant_digits = "-1"

prompt> set_app_var report_default_significant_digits 4
4

prompt> printvar report_default_significant_digits
report_default_significant_digits = "4"

prompt> set_app_var report_default_significant_digits -44
Error: can't set report_default_significant_digits: must be in range 0 to
 13
        Use error_info for more info. (CMD-013)
4

prompt> printvar report_default_significant_digits
report_default_significant_digits = "4"
```

See Also

- report_cell
- report_clock_gating_check
- report_constraint
- report_net
- report_path_budget
- report_qor
- report_timing
- write_sdf

Feedback

report_timing_use_accurate_delay_symbol

Specifies whether to use accurate delay symbols in *report_timing* reports.

Data Types

Boolean

Default true

Description

This variable specifies whether to use accurate delay symbols in the "Incr" column of reports generated by *report_timing*. When the variable is set to true (the default), the following symbols are used:

```
&: Delay information is timing-annotated using Elmore delay calculation
c: Delay information is from Arnoldi calculation with accurate CCS
@: Delay information is from Arnoldi calculation
a: Delay information is from postroute AWE calculation
w: Delay information is from preroute AWE calculation
```

When the variable is set to false, the "c", "@" and "a" symbols are not used; preroute delay increments are marked with "*" and postroute delay increments are marked with "&". The asterisk symbol (*) means back-annotation using preroute Elmore extraction or SDF.

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

See Also

- report_timing

rom_auto_inferring

Infers ROM from the RTL description.

Data Types

Boolean

Default true in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Feedback

Description

When this variable is set to *true*, the tool attempts to infer ROM from the RTL description. When set to *false*, no attempt is made to infer ROM from the RTL description.

See Also

- compile

---

route_guide_naming_style

Specifies the naming convention to be used by the *create_route_guide* command.

Data Types

string

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

%s_%d in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

This variable specifies the naming convention to be used by the *create_route_guide* command. The variable string must contain only one %s (percent s) and one %d (percent d) character sequence. To use a percent sign in the route guide name, two are needed in the string (%%).

See Also

- create_route_guide

---

rtl_load_resistance_factor

Specifies a factor to be used by the *set_rtl_load* command to calculate resistance values from capacitance values for RTL loads.

Data Types

float

Default 0

Description

This variable specifies a factor to be used by the *set_rtl_load* command to calculate resistance values from capacitance values for RTL loads.

Feedback

You do not need to specify resistance values directly to *set_rtl_load*. Instead, you can cause the resistance value to be calculated as a constant factor times capacitance by setting the *rtl_load_resistance_factor* variable to the constant factor required. Then, you execute *set_rtl_load* with only the *-capacitance* option, and the command calculates the resistance from the specified capacitance using the constant factor.

For example, if you set the *rtl_load_resistance* variable to 0.5, specify the rtl-load capacitance of a pin as 4, and do not specify the rtl-load resistance, *set_rtl_load* calculates the rtl-load resistance to be 2. The factor is applied to each annotated pin of a net individually, not to the net's capacitance as a whole. Note that the default library units are used throughout.

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

See Also

- remove_rtl_load
- set_rtl_load

S

sdc_runtime_analysis_enable

Issues information messages to identify and fix runtime issues related to SDC constraints. This might impact quality of reports (QoR) as SDC constraints might be relaxed.

Data Types

Boolean

Default false

Description

This variable helps you to debug only SDC issues in your design that result in long runtime. If you suspect long runtime is due to SDC constraints, set the *sdc_runtime_analysis_enable* variable to *true* to identify and debug runtime issues related to SDC constraints. The tool

- Identifies and reports SDC issues
- Generates a Tcl file with potential fixes for the identified SDC issues by relaxing a few constraints.

You must review the Tcl file for your design intent and source the Tcl file during the next compile session after setting the *sdc_runtime_analysis_enable* variable to *false* to see improvements in the runtime of the tool.

You can get an estimate of runtime benefit after applying the SDC fixes in the same debug run (instead of sourcing the file during the next compile session). This feature is available in the Design Compiler NXT tool.

The Design Compiler NXT tool automatically applies the potential fixes (provided in the Tcl file) in the same debug run. There might be an impact on QoR when you enable automatic fixing of SDC issues in the Design Compiler NXT tool due to the change in SDC constraints during the same debug run.

Note: Do not keep the *sdc_runtime_analysis_enable* variable always set to *true*. Use the variable only to debug runtime issues related to SDC constraints.

To avoid the Design Compiler NXT tool from automatically applying the fixes provided in the Tcl file for SDC issues, set the *sdc_runtime_fixing_enable* variable to *false*. The default is *true*.

```
dc_shell-topo> set_app_var sdc_runtime_analysis_enable true
```

For more details about the different SDC constraints identified and the potential fixes applied by the tool, see the "Reporting Runtime Issues Related to SDC Constraints" section in the Design Compiler User Guide.

The Design Compiler NXT and Design Compiler Graphical tools by default generate a Tcl script (sdc_runtime_<timestamp>.log.tcl) with the information message and Tcl commands to disable the identified SDC constraints that cause runtime issues. For example, sdc_runtime_24_10_2019_02_11_20.log.

To specify a different log file name or a standard output file, use the *sdc_runtime_analysis_log_file* variable.

The following variable creates a new log file name to redirect the output to the my_log_file.log file and to redirect Tcl file to the my_log_file.log.tcl file:

```
prompt> set_app_var sdc_runtime_analysis_log_file my_log_file.log
```

The following variable redirects log output to sdc_runtime<timestamp>.log and redirects Tcl file to sdc_runtime<timestamp>.log.tcl:

```
prompt> set_app_var sdc_runtime_analysis_log_file
```

The following variable redirects log to a standard output console and redirects Tcl file to sdc_runtime<timestamp>.log.tcl

```
prompt> set_app_var sdc_runtime_analysis_log_file ""
```

Feedback

See Also

- sdc_runtime_analysis_log_file
- sdc_runtime_nets_missing_exceptions_fanout_threshold
- sdc_runtime_top_fanout_nets_missing_exceptions
- sdc_runtime_paths_missing_inter_clock_constraints
- sdc_runtime_port_clock_constraint_threshold
- sdc_runtime_tightly_constrained_path_group_slack_percentage
- sdc_runtime_tightly_constrained_same_clock_path_groups
- sdc_runtime_hier_block_pins_timing_path_threshold
- sdc_runtime_hier_block_pins_top_timing_paths
- sdc_runtime_unused_clocks_threshold
- sdc_runtime_fixing_enable

---

sdc_runtime_analysis_log_file

Redirects logs to a file.

Data Types

String

Default sdc_runtime.log

Description

This variable is used to redirect logs to a file. Note that you must set the *sdc_runtime_analysis_enable* variable to true to use this variable.

If a file name is not specified, the information messages are stored in the sdc_runtime.log file. To specify a new log file name, use the variable as follows:

set_app_var sdc_runtime_analysis_enable true

set_app_var sdc_runtime_analysis_log_file my_log_file.log - a new log file name to redirect it to a my_log_file.log

set_app_var sdc_runtime_analysis_log_file - redirects log to sdc_runtime.log

set_app_var sdc_runtime_analysis_log_file "" - redirects log to standard output console

Feedback

See Also

- sdc_runtime_analysis_enable

---

sdc_runtime_fixing_enable

Fixes the runtime issues related to SDC constraints, identified when sdc_runtime_analysis_enable switch is enabled.

Data Types

Boolean

Default true

Description

This variable controls the dirty constraints fixes, which helps in improving runtime. You can disable this variable if automatic fixing of dirty constraints is not needed. Enabling SDC fixing can change the QoR trajectory in the compile run. The fixed constraints are written out to a Tcl file to support customization.

You can use the tcl commands in your tcl script to review the fixes. By default, the *sdc_runtime_fixing_enable* variable is set to true. To set the variable to false,

dc_shell-topo> *set_app_var sdc_runtime_fixing_enable false*

If a file name is not specified, the tcl commands are stored in the sdc_runtime_<timestamp>.log.tcl file. To specify a file name, use the *sdc_runtime_analysis_log_file* variable. To redirect the messages to a tcl file, use the variable as follows:

set_app_var sdc_runtime_analysis_log_file my_log_file.log - a new log file name to redirect it to a my_log_file.log.tcl

set_app_var sdc_runtime_analysis_log_file - redirects tcl to sdc_runtime<timestamp>.log.tcl

set_app_var sdc_runtime_analysis_log_file "" - also redirects tcl to sdc_runtime<timestamp>.log.tcl

See Also

- sdc_runtime_analysis_enable
- sdc_runtime_analysis_log_file
- sdc_runtime_nets_missing_exceptions_fanout_threshold
- sdc_runtime_top_fanout_nets_missing_exceptions

Feedback

- sdc_runtime_paths_missing_inter_clock_constraints
- sdc_runtime_port_clock_constraint_threshold
- sdc_runtime_tightly_constrained_path_group_slack_percentage
- sdc_runtime_tightly_constrained_same_clock_path_groups

sdc_runtime_hier_block_pins_timing_path_threshold

Controls the timing arcs threshold and the number of pins to display when the design has hierarichal blocks.

Data Types

Integer

Default 500

Description

When the tool spends huge time during buffering, this could be due to high number of timing arcs through the hierarichal block pins in the design.

The variable controls threshold value of number of timing arcs for displaying the TIM-609 message.

By default, the tool prints this message only when number of timing arcs through hierarchical pin exceeds 500 in a design.

The variable setting is effective only when you set the *sdc_runtime_analysis_enable* variable to true.

See Also

- sdc_runtime_hier_block_pins_top_timing_paths
- sdc_runtime_analysis_enable

sdc_runtime_hier_block_pins_top_timing_paths

Controls the display of number of pins with high timing arcs when the design has hierarichal blocks.

Data Types

Integer

Default 100

Feedback

Description

When the tool spends huge time during buffering, this could be due to high number of timing arcs through the hierarichal block pins in the design.

The variable controls the display of hierarichal pins with high number of timing arcs under TIM-609 message.

By default, the tool prints this message only for the top 100 such pins in a design.

The variable setting is effective only when you set the *sdc_runtime_analysis_enable* variable to true.

See Also

- sdc_runtime_hier_block_pins_timing_path_threshold
- sdc_runtime_analysis_enable

---

sdc_runtime_nets_missing_exceptions_fanout_threshold

Controls the fanout threshold and the number of nets to display when high-fanout nets are not set with the *set_ideal_network* or *set_dont_touch* command.

Data Types

Integer

Default 500

Description

When the tool spends huge time during buffering, this could be due to high-fanout nets in the design. In such cases, the timing complexity could be because of the high-fanout nets that are not set with the ideal_network or dont_touch constraint.

For more information, see the TIM-606 information message. By default, the variable prints this message for only top 100 paths with fanout > 100 that are not set with the *set_ideal_network* or *set_dont_touch* command.

The variable controls the fanout threshold and the number of nets to display when high-fanout nets are not set with the ideal_network or dont_touch constraints.

The variable setting is effective only when you set the *sdc_runtime_analysis_enable* variable to true.

Feedback

See Also

- sdc_runtime_top_fanout_nets_missing_exceptions
- sdc_runtime_analysis_enable

---

sdc_runtime_paths_missing_inter_clock_constraints

Controls the number of paths to display when inter-clock constraints are missing for each path group.

Data Types

Integer

Default 5

Description

If the worst net slack (WNS) printed is high in a design log file, the reason could be missing multicycle or false path constraint. This is because you missed applying multicycle or false path constraints. Optimizing the design in such cases could increase runtime.

For more information, see the TIM-608 information message. By default, the variable prints this message for only 5 paths for each path group with tightly constrained paths that do not have false or multicycle path.

The variable controls the number of paths to display when inter-clock constraints are missing for each path group. The variable setting is effective only when you set the *sdc_runtime_analysis_enable* variable to true.

See Also

- sdc_runtime_analysis_enable

---

sdc_runtime_port_clock_constraint_threshold

Controls the number of paths to display for each port.

Data Types

Integer

Default 20

Feedback

Description

When there are too many clocks associated with input and output pins, it can mean that there are multiple timing paths through a pin. Optimizing the design in such cases could increase runtime. The variable controls the number of paths to display for each port.

For more information, see the TIM-607 information message. By default, the variable prints this message only when the number of paths on a port exceed 20.

The variable setting is effective only when you set the *sdc_runtime_analysis_enable* variable to true.

See Also

- sdc_runtime_analysis_enable

---

sdc_runtime_tightly_constrained_path_group_slack_percentage

Sets the threshold to determine whether a path is tightly constrained.

Data Types

Integer

Default 75

Description

The variable sets the threshold for determining whether a path is tightly constrained. It considers paths with critical slack more than 75 percent of a clock period.

For more information, see the TIM-602 information message. By default, the variable prints this message only when paths with slack more than 75 percent of a clock period are tightly constrained.

The variable setting is effective only when you set the *sdc_runtime_analysis_enable* variable to true.

---

sdc_runtime_tightly_constrained_same_clock_path_groups

Sets the threshold to identify whether too many path groups use the same clock.

Data Types

Integer

Default 50

Feedback

Description

This variable is a threshold for identifying whether there are too many path groups using the same clock.

For more information, see the TIM-603 information message. By default, the variable prints this message only when more than 50 path groups are created with a single clock.

The variable setting is effective only when you set the *sdc_runtime_analysis_enable* variable to true.

See Also

- sdc_runtime_analysis_enable

---

sdc_runtime_top_fanout_nets_missing_exceptions

Controls the fanout threshold and the number of nets to display when high-fanout nets are not set with the *set_ideal_network* or *set_dont_touch* command.

Data Types

Integer

Default 200

Description

The variable controls the fanout threshold and the number of nets to display when high-fanout nets that are not set with the ideal_network or dont_touch constraints.

For more information, see the TIM-606 information message. By default, the variable prints this message for only top 100 paths with fanout > 100 that are not set with the *set_ideal_network* or *set_dont_touch* command.

The variable setting is effective only when you set the *sdc_runtime_analysis_enable* variable to true.

See Also

- sdc_runtime_analysis_enable
- sdc_runtime_nets_missing_exceptions_fanout_threshold

---

sdc_runtime_unused_clocks_threshold

Controls the minimum number of unused clocks required in the design, to enable display of TIM-601 messages

Feedback

Data Types

Integer

Default 5

Description

Tool can take extra runtime to work on unused clocks in the design. In such cases, it is better to remove such clocks to save time.

By default, the tool prints the TIM-601 message only when the design has more than 5 unused clocks. Use this variable to control this threshold.

The variable setting is effective only when you set the *sdc_runtime_analysis_enable* variable to true.

See Also

- sdc_runtime_top_fanout_nets_missing_exceptions
- sdc_runtime_analysis_enable

---

sdc_write_unambiguous_names

Ensures that cell, net, pin, lib_cell, and lib_pin names that are written to the SDC file are not ambiguous.

Data Types

Boolean

Default true

Description

This variable ensures that cell, net, pin, lib_cell, and lib_pin names that are written to the SDC file are not ambiguous.

When the hierarchy has been partially flattened, embedded hierarchy separators can make names ambiguous. It is unclear which hierarchy separator characters are part of the name and which are real separators.

Beginning with SDC Version 1.2, hierarchical names can be made unambiguous using the *set_hierarchy_separator* SDC command and/or the *-hsc* option for the *get_cells*, *get_lib_cells*, *get_lib_pins*, *get_nets*, and *get_pins* SDC object access commands. By default, the tools write an SDC file, using these features to create unambiguous names.

It is wise to write SDC files that contain names that are not ambiguous. However, if you are using a third-party application that does not fully support SDC 1.2 or later versions

(that is, it does not support the unambiguous hierarchical names features of SDC), you can suppress these features by setting the variable *sdc_write_unambiguous_names* to *false*. The *write_sdc* command issues a warning if you have set this variable to *false*.

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

See Also

- printvar
- write_sdc

---

sdfout_allow_non_positive_constraints

Writes out PATHCONSTRAINT constructs with nonpositive (<= 0) constraint values. This variable will be obsolete in the next release. Please adjust your scripts accordingly.

Data Types

Boolean

Default false in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Writes out PATHCONSTRAINT constructs with nonpositive (<= 0) constraint values. When *true*, *write_constraints -format sdf* writes out PATHCONSTRAINT constructs with nonpositive (<= 0) constraint values. When *false* (the default), paths with nonpositive constraints are written with a constraint value of 0.01.

Nonpositive constraints can occur when the arrival time at a path startpoint is larger than the required arrival time at the path endpoint. This typically indicates an error, but is sometimes valid when generating constraints for a subdesign.

To determine the current value of this variable, type *printvar sdfout_allow_non_positive_constraints*. For a list of all *links_to_layout* variables and their current values, type *print_variable_group links_to_layout*.

---

sdfout_min_fall_cell_delay

Specifies the minimum non-back-annotated fall cell delay that the *write_timing* command writes to a timing file in SDF format. This variable will be obsolete in the next release. Please adjust your scripts accordingly.

Feedback

Data Types

float

Default 0 in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Specifies the minimum non-back-annotated fall cell delay that the *write_timing* command writes to a timing file in SDF format. The value of this variable can be *positive*, *negative*, or *zero* (the default); the unit must be the same as the timing unit in the technology library.

By default, if this variable is not set, *write_timing* writes to the SDF file all fall cell delay values greater than or equal to zero. You can override this default behavior for non-back-annotated delays by setting *sdfout_min_fall_cell_delay* to a minimum value; *write_timing* does not write values that are less than this minimum. However, you cannot override the default behavior for back-annotated delays; *write_timing* always writes values of delays that have been back-annotated, regardless of the value of this variable.

Use this variable to filter non-back-annotated delays so that *write_timing* writes only values that are significant (greater than the specified minimum value). Also, if you do not want non-annotated fall cell delays to be written to the SDF file, set this variable to a value higher than any of the non-annotated cell delays in the design.

See Also

- sdfout_min_rise_cell_delay

---

sdfout_min_fall_net_delay

Specifies the minimum non-back-annotated fall net delay that *write_timing* can write to a timing file in SDF format. This variable will be obsolete in the next release. Please adjust your scripts accordingly.

Data Types

float

Default 0 in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Feedback

Description

Specifies the minimum non-back-annotated fall net delay that *write_timing* can write to a timing file in SDF format. The value of this variable can be *positive*, *negative*, or *zero* (the default); the unit must be the same as the timing unit in the technology library.

By default, if this variable is not set, *write_timing* writes to the SDF file all fall net delay values greater than or equal to zero. You can override this default behavior for non-back-annotated delays by setting *sdfout_min_fall_net_delay* to a minimum value; *write_timing* does not write values that are less than this minimum. However, you cannot override the default behavior for back-annotated delays; *write_timing* always writes values of delays that have been back-annotated, regardless of the value of this variable.

Use this variable to filter non-back-annotated delays so that *write_timing* writes only values that are significant (greater than the specified minimum value). Also, if you do not want any non-annotated fall net delays to be written to the SDF file, set this variable to a value higher than any of the non-annotated net delays in the design.

See Also

- sdfout_min_rise_net_delay

---

sdfout_min_rise_cell_delay

Specifies the minimum non-back-annotated rise cell delay that *write_timing* can write to a timing file in SDF format. This variable will be obsolete in the next release. Please adjust your scripts accordingly.

Data Types

float

Default 0 in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Specifies the minimum non-back-annotated rise cell delay that *write_timing* can write to a timing file in SDF format. The value of this variable can be *positive*, *negative*, or *zero* (the default); the unit must be the same as the timing unit in the technology library.

By default, if this variable is not set, *write_timing* writes to the SDF file all rise cell delay values greater than or equal to zero. You can override this default behavior for non-back-annotated delays by setting *sdfout_min_rise_cell_delay* to a minimum value; *write_timing* will not write values less than this minimum. However, you cannot override the default

behavior for back-annotated delays; *write_timing* always writes values of delays that have been back-annotated, regardless of the value of this variable.

Use this variable to filter non-back-annotated delays so that *write_timing* writes only values that are significant (greater than the specified minimum value). Also, if you do not want any non-annotated rise cell delays to be written to the SDF file, set this variable to a value higher than any of the non-annotated cell delays in the design.

See Also

- sdfout_min_fall_cell_delay

---

sdfout_min_rise_net_delay

Specifies the minimum non-back-annotated rise net delay that the *write_timing* command can write to a timing file in SDF format. This variable will be obsolete in the next release. Please adjust your scripts accordingly.

Data Types

float

Default 0 in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Specifies the minimum non-back-annotated rise net delay that the *write_timing* command can write to a timing file in SDF format. The value of this variable can be *positive*, *negative*, or *zero* (the default); the unit must be the same as the timing unit in the technology library.

By default, if this variable is not set, *write_timing* writes to the SDF file all rise net delay values greater than or equal to zero. You can override this default behavior for non-back-annotated delays by setting *sdfout_min_rise_net_delay* to a minimum value; *write_timing* does not write values less than this minimum. However, you cannot override the default behavior for back-annotated delays; *write_timing* always writes values of delays that have been back-annotated, regardless of the value of this variable.

Use this variable to filter non-back-annotated delays so that *write_timing* writes only values that are significant (greater than the specified minimum value). Also, if you do not want any non-annotated rise net delays to be written to the SDF file, set this variable to a value higher than any of the non-annotated net delays in the design.

Feedback

See Also

- sdfout_min_fall_net_delay

---

sdfout_time_scale

Specifies the time scale of the delays written to timing files in SDF format. This variable will be obsolete in the next release. Please adjust your scripts accordingly.

Data Types

float

Default 1 in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Specifies the time scale of the delays written to timing files in SDF format. Delays from Design Compiler are written to timing files with *write_timing*. The *sdfout_time_scale* variable must be set if the library has no time unit specified and if the time unit of the delays in the library is different than 1 nanosecond. By default, the time unit is nanosecond and the time scale is 1. The only valid values for the SDF format are 0.001, 0.01, 0.1, 1, 10 and 100. The time unit is specified in the library with the attributes *time_scale* and *time_unit_name*.

For example, a library with timing values in 10 picoseconds is specified with the attributes:

time_scale = 10 and time_unit_name = ps

When the attribute *time_scale* is missing in the library, use the *sdfout_time_scale* variable to specify the scale of the timing unit. For example, if the library has no time unit specified but is in 10 ns, set *sdfout_time_scale* to 10 in order to specify the time unit as 10 ns.

---

sdfout_top_instance_name

Specifies the name prepended to all instance names when writing timing files in SDF format. This variable will be obsolete in the next release. Please adjust your scripts accordingly.

Data Types

string

Default "" in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Feedback

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Specifies the name prepended to all instance names when writing timing files in SDF format. Timing files are written with the *write_timing* command. By default *write_timing* prepends no name to all cell instance names. Set this variable when you want the cell instance names to contain a prepended name.

For example set *sdfout_top_instance_name* = *"stim.cell1"* if the timing file should contain:

```
(DESIGN "fifo")
(DIVIDER .)
(CELL
   (CELLTYPE "fifo")
   (INSTANCE stim\\.cell1)
)
(CELL
   (CELLTYPE "AND2")
   (INSTANCE stim\\.cell1.U1)
```

With the previous example, if *sdfout_top_instance_name* = *""* timing file will contain:

```
(DESIGN "fifo")
(DIVIDER .)
(CELL
   (CELLTYPE "fifo")
   (INSTANCE )
)
(CELL
   (CELLTYPE "AND2")
   (INSTANCE U1)
```

sdfout_write_to_output

Specifies whether the *write_timing -f sdf* command writes interconnect delays between cells and top-level output ports. This variable will be obsolete in the next release. Please adjust your scripts accordingly.

Data Types

Boolean

Default false in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Feedback

Description

Specifies whether the *write_timing -f sdf* command writes interconnect delays between cells and top-level output ports. The *sdfout_write_to_output* variable also determines whether output to output pin IOPATH statements are written for cells that contain output-to-output timing arcs. v1.0 SDF does not support output-to-output timing for either IOPATH or INTERCONNECT statements. However, the Synopsys Simulator does support output-to-output timing for these statements.

Set this variable to *true*, if the targeted SDF reader is the Synopsys Simulator.

Leave this variable set to *false*, the default, to ensure that generated SDF files comply with the v1.0 specification.

Set this variable before using *write_timing*. To check the value of this variable, use the command *printvar sdfout_write_to_output*.

---

search_path

Specifies directories that the tool searches for files specified without directory names.

Data Types

list

Default {*search_path* + .}

Description

This variable specifies directories that the tool searches for files specified without directory names. The search includes looking for technology and symbol libraries, design files, and so on. The value of this variable is a list of directory names and is usually set to a central library directory.

To determine the current value of this variable, use the *printvar search_path* command. For a list of all system variables and their current values, use the *print_variable_group system* command.

---

seqmap_prefer_registers_with_multibit_equivalent

Controls the *compile_ultra* command for preferentially mapping sequential cells to single-bit registers with compatible multibit registers.

Data Types

Boolean

Default false

Feedback

Description

When this variable is set to *true*, the *compile_ultra* command tries to map the registers using single-bit registers with equivalent multibit registers. The goal is to improve the multibit register's packing ratio of the design.

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

For a list of all compile variables and their current values, use the *print_variable_group compile* command.

See Also

- hdlin_infer_multibit

---

sh_allow_tcl_with_set_app_var

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

Data Types

string

Default true

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
- set_app_var

---

sh_allow_tcl_with_set_app_var_no_message_list

Suppresses CMD-104 messages for variables in this list.

Data Types

string

Feedback

Default ""

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
- set_app_var

---

sh_arch

Indicates the system architecture of your machine.

Data Types

string

Default linux64

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_sdp

Variable to enable automatic SDP data collection

Data Types

Boolean

Default false

Description

Set this variable to True if auto SDP data collection is desired. The commands to be measured with SDP must be added to sh_auto_sdp_commands

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

```
prompt> printvar sh_auto_sdp
```

Feedback

See Also

- sh_auto_sdp_commands

---

sh_auto_sdp_commands

Variable that stores the commands' names for which SDP data will be collected.

Data Types

List of string

Default ""

Description

List of commands' names for which SDP data will be collected when auto SDP is enabled. The first time one of these commands is called, SDP will be started. Further commands will be labeled in SDP report if they are on this list.

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

```
prompt> printvar sh_auto_sdp_commands
```

See Also

- sh_auto_sdp

---

sh_auto_sdp_crte_timeperiod

Variable containing the time period in seconds for collecting the CRTE.

Data Types

Int

Default 0

Description

Timeperiod in seconds to collect CRTE info for SDP.

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

```
prompt> printvar sh_auto_sdp_crte_timeperiod
```

See Also

- sh_auto_sdp

Feedback

sh_auto_sdp_delete

Variable to enable past SDP data deletion.

Data Types

Boolean

Default false

Description

Set to true to delete past SDP.tgz files in current folder (enumerated SDP files, if any, will not be deleted.)

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

```
prompt> printvar sh_auto_sdp_delete
```

See Also

- sh_auto_sdp

sh_auto_sdp_stack_trace_frequency

Variable containing the stack trace frequency in tenths of a second for collecting the CRTE.

Data Types

Int

Default 0

Description

Frequency in tenths of a second to collect stack trace info for SDP.

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

```
prompt> printvar sh_auto_sdp_stack_trace_frequency
```

See Also

- sh_auto_sdp

sh_auto_sdp_verbose

Variable to enable verbose operation for auto SDP.

Feedback

Data Types

Boolean

Default false

Description

Enable this variable to activate verbose on auto SDP usage.

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

```
prompt> printvar sh_auto_sdp_verbose
```

See Also

- sh_auto_sdp

---

sh_command_abbrev_mode

Sets the command abbreviation mode for interactive convenience.

Data Types

string

Default Anywhere

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
- get_app_var
- set_app_var

Feedback

sh_command_abbrev_options

Turns off abbreviation of command dash option names when false.

Data Types

Boolean

Default true

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
- get_app_var
- set_app_var

sh_command_log_file

Specifies the name of the file to which is written a log of the initial values of variables and executed commands.

Data Types

string

Default command.log

Description

Specifies the name of the file to which is written a log of the initial values of variables and executed commands.

Feedback

By default, the tool writes the log to a file named command.log. If the value is an empty string, a command log file is not created.

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

See Also

- printvar
- view_command_log_file
- view_log_file

---

sh_continue_on_error

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

Data Types

Boolean

Default true

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
- set_app_var
- source
- sh_script_stop_severity

Feedback

sh_deprecated_is_error

Raise a Tcl error when a deprecated command is executed.

Data Types

Boolean

Default false

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
- set_app_var

sh_dev_null

Indicates the current null device.

Data Types

string

Default /dev/null

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

sh_disabled_is_error

Raise a Tcl error when a disabled command is executed.

Data Types

Boolean

Default true

Feedback

Description

When set this variable causes a Tcl error to be raised when a disabled command is executed. When false, only a warning message is issued.

Disabled commands have no effect.

See Also

- get_app_var
- set_app_var

---

sh_enable_machine_monitor

When set to true, prints resource report and enables further free resources reports during Compile

Data Types

Boolean

Default true

Description

When set to true, It will print a full report on resources, with details of the current build, CPU, RAM, Swap and Disk. It will also enable a smaller report on free resources and CPU load during intermediate stages of Compile.

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

```
prompt> printvar sh_enable_machine_monitor
```

---

sh_enable_stdout_redirect

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

Data Types

Boolean

Default true

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.

Feedback

See Also

- get_app_var
- set_app_var

---

sh_help_shows_group_overview

Changes the behavior of the "help" command.

Data Types

string

Default true

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
- set_app_var

---

sh_language

Controls the tool language used by the application interactive console.

Data Types

string

Default tcl

Description

This variable controls the language used by the application interactive console. Some applications support languages in addition to Tcl as an extension language.

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

Feedback

See Also

- get_app_var
- set_app_var

---

sh_new_variable_message

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

Data Types

Boolean

Default false

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_varcommand 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
- set_app_var
- sh_new_variable_message_in_proc
- sh_new_variable_message_in_script

Feedback

sh_new_variable_message_in_proc

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

Data Types

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
- set_app_var
- sh_new_variable_message
- sh_new_variable_message_in_script

sh_new_variable_message_in_script

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

Feedback

Data Types

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>
```

Feedback

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
- set_app_var
- sh_new_variable_message
- sh_new_variable_message_in_proc

---

sh_obsolete_is_error

Raise a Tcl error when an obsolete command is executed.

Data Types

Boolean

Default false

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.

Feedback

See Also

- get_app_var
- set_app_var

---

sh_output_log_file

This variable is used to name the file to record console output. Set the variable to the empty string to disable output capture.

Data Types

String

Default ""

Description

This variable can be used to save almost all console output to a log file. The log file is useful for bug reproduction and reporting.

The first time the variable is set to a valid filename, all previously logged output is written to the specified file, overwriting any previous contents of the file. All subsequent logged output is appended to the file unless the variable is later reset.

If the variable is later changed to an empty string, all logged output is disabled.

If the variable is later set to an non-empty string that is an invalid filename, the new value is ignored and the old value is restored.

If the variable is later set to an non-empty string that is an valid filename, all subsequent logged output is written to this file. Any previous contents of the file are overwritten.

The log file may not capture the banner text from the beginning of the session, and does not capture the stack trace that is printed following a fatal error. You can use the UNIX tee command or redirect the output to capture this information.

This variable is not available in de_shell mode.

See Also

- printvar

---

sh_product_version

Indicates the version of the application currently running.

Feedback

Data Types

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

---

sh_script_stop_severity

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

Data Types

string

Default None

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.

Feedback

See Also

- get_app_var
- set_app_var
- source
- sh_continue_on_error

---

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.

Data Types

string

Default None

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
- set_app_var
- source

Feedback

- sh_continue_on_error
- sh_script_stop_severity
- CMD-081
- CMD-082

---

sh_source_logging

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

Data Types

Boolean

Default true

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
- set_app_var
- source

---

sh_source_uses_search_path

Causes the *search* command to use the *search_path* variable to search for files. This variable is for use in *dc_shell-t* (Tcl mode of dc_shell) only.

Data Types

Boolean

Default true

Feedback

Description

Causes the *search* command to use the *search_path* variable to search for files, when *sh_source_uses_search_path* is *true* (the default). When *false*, the *source* command considers this variable's file argument literally. This variable is for use in *dc_shell-t* (Tcl mode of dc_shell) only.

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

See Also

- printvar
- source
- search_path

---

sh_tcllib_app_dirname

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

Data Types

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

---

sh_user_man_path

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

Data Types

list

Default ""

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

Feedback

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
- get_app_var
- man
- set_app_var

---

si_ccs_use_gate_level_simulation

Affects the behavior of timing analysis and postroute optimization when crosstalk effects are enabled.

Data Types

Boolean

Default false

Group

si_variables

Description

This variable affects the behavior of timing analysis and postroute optimization when crosstalk effects are enabled.

When set to *true*, it enables the use of CCS noise engine for delta delay computation. To use this feature, make sure that your library contains proper CCS noise data.

Feedback

Enabling this feature improves the IC Compiler and PrimeTime SI correlation, but it can lead to runtime increase.

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

See Also

- report_timing

---

si_max_parallel_computations

Sets the degree of parallelism used for parallel signal integrity computations.

Data Types

integer

Default 0

Description

This application variable specifies the degree of parallelism used for parallel signal integrity computations.

When the variable has a value of *1*, parallel signal integrity computation is disabled.

When the variable has a value of *0* (the default), or a value that is larger than *1*, parallel signal integrity computation is enabled.

See Also

- set_host_options

---

si_xtalk_composite_aggr_noise_peak_ratio

Controls the composite aggressor selection for crosstalk analysis.

Data Types

float

Default 0.01

Description

Specifies the threshold value in crosstalk bump to VDD ratio, below which aggressors are selected into the composite aggressor group. The default is *0.01*, which means all the aggressor nets with crosstalk bump to VDD ratio less than *0.01*

Feedback

is selected into the composite aggressor group. This variable works together with other filtering threshold variables, *si_filter_per_aggr_noise_peak_ratio* and *si_filter_accum_aggr_noise_peak_ratio*, to determine which aggressors can be selected into the composite aggressor group.

To determine the current value of this variable, use the following command:

```
printvar si_xtalk_composite_aggr_noise_peak_ratio
```

See Also

- si_xtalk_composite_aggr_quantile_high_pct

---

si_xtalk_composite_aggr_quantile_high_pct

Controls the composite aggressor creation for statistical analysis.

Data Types

float

Default 99.73

Description

Sets the desired probability in percentage format that any given real combined bump height is less than or equal to the computed composite aggressor bump height. Given the desired probability, the resulting quantile value for the composite aggressor bump height is calculated.

The default of this variable is *99.73*, which corresponds to a 3-sigma probability that the real bump height from any randomly-chosen combination of aggressors is covered by the composite aggressor bump height.

To determine the current value of this variable, use the following command:

```
prompt> printvar si_xtalk_composite_aggr_quantile_high_pct
```

See Also

- si_xtalk_composite_aggr_noise_peak_ratio

---

simplified_verification_mode

Performs optimization so that formal verification is prioritized over quality of results (QoR).

Feedback

Data Types

Boolean

Default false

Group

none

Description

Setting this variable to *true* adjusts optimization inside the tool to prioritize formal verification compatibility over QoR. This enables single-pass verification for formal verification to pass with as little effort as possible. The variable is set to *false* by default.

This variable affects the optimization done during the reading and elaboration of RTL files and when the *insert_clock_gating* and *compile_ultra* commands are run.

When you set this variable to *true*, the *compile_ultra* command runs with the *-no_autoungroup* option enabled. However, the *-retime* option of the *compile_ultra* command is ignored, and the *-global* option of the *insert_clock_gating* and *replace_clock_gates* commands is ignored.

The tool sets the value for the following environment variables when the *simplified_verification_mode* variable is set to *true* regardless of the value you specify:

```
compile_ultra_ungroup_dw = false,
      compile_clock_gating_through_hierarchy = false
      hdlin_verification_priority = true
```

The value for these variables is restored to the user-specified value when you set the *simplified_verification_mode* variable to *false*. If you do not specify the value for these variables, the tool uses the default value.

When you run the *compile_ultra* command, the *simplified_verification_mode* variable adjusts the optimizations on datapath logic. It also identifies CRC logic in the design and isolates it by creating a new hierarchy.

When you set the *simplified_verification_mode* variable to *true*, designs with the *optimize_registers* attribute and DW_div_pipe components are not retimed because they can adversely affect verification success. However, optimizing these designs without retiming them can have a large impact on QoR and runtime. To enable the retiming of these designs, set the *simplified_verification_mode_allow_retiming* variable to *true* when you enable the *simplified_verification_mode* variable.

See Also

- simplified_verification_mode_allow_retiming

Feedback

simplified_verification_mode_allow_retiming

Controls whether *DW_div_pipe* components and designs with the *optimize_registers* attribute are retimed when the *simplified_verification_mode* variable is set to *true*.

Data Types

Boolean

Default false

Group

none

Description

When you set the *simplified_verification_mode* variable to *true*, designs with the *optimize_registers* attribute and DW_div_pipe components are not retimed because they can adversely affect verification success. However, optimizing these designs without retiming them can have a large impact on QoR and runtime. To enable the retiming of these designs, set the *simplified_verification_mode_allow_retiming* variable to *true* when you enable the *simplified_verification_mode* variable.

See Also

- simplified_verification_mode

single_group_per_sheet

Specifies to the tool to put only 1 logic group on a sheet.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable specifies to the tool to put only 1 logic group on a sheet. Using this partitioning option set to *true* eliminates the possibility of more than 1 off-sheet connector with the same name being on a single sheet. The default value is *false*.

Feedback

site_info_file

Contains the path to the site information file for licensing.

Data Types

string

Default ""

Description

This variable contains the path to the site information file for licensing. The default value is the empty string.

sort_outputs

Sorts output ports on the schematic by port name.

Data Types

string

Default false

Description

This variable sorts output ports on the schematic by port name.

spg_auto_ndr_net_length_threshold

Restricts the length of the nets assigned nondefault routing rules during optimization.

Data Types

Integer

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

100 in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Group

syn-qor

Feedback

Description

Sets a minimum length threshold on nets that are considered for nondefault routing rules during optimization in the Design Compiler Graphical tool. The value you specify is multiplied by the minimum height of the unit tile.

For example, if you specify a threshold value of 100, the minimum length of the nets considered for nondefault routing rule assignment is 100 times the minimum height of the unit tile.

See Also

- compile_ultra

---

spg_congestion_placement_in_incremental_compile

Enables congestion-driven placement in incremental compile to improve congestion while preserving quality of results.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode and Design Compiler NXT topographical mode

Description

When you set the *spg_congestion_placement_in_incremental_compile* variable to *true* before running *compile_ultra -incremental*, the tool runs congestion-driven placement during incremental compilation to improve congestion. This variable also improves congestion correlation between post-incremental compile and post *place_opt* results.

You do not need to enable Zroute to use the variable. If Zroute is enabled, the tool runs congestion-driven placement in incremental compile by default.

Examples

The following example shows how to enable and disable congestion-driven placement in incremental compile for a design:

```
prompt> set_app_var spg_congestion_placement_in_incremental_compile true
prompt> set_app_var spg_congestion_placement_in_incremental_compile false
```

Feedback

See Also

- compile_ultra
- set_attribute
- get_attribute
- remove_attribute

---

spg_enable_multithreaded_zroute

Enables multicore Zroute during compile passes which optionally use the Zroute global router.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode and DC Explorer

true in Design Compiler NXT topographical mode

Description

This feature is available only in Design Compiler-NXT.

This variable enables deterministic multicore Zroute during compile passes which use the Zroute global routing. Zroute is called by 'compile_ultra -spg' under certain higher-effort settings, such as placer_enable_enhanced_router, and set_ahfs_options -global_route. not used under default compile settings.

When set to *true* (the default), if Zroute is run during compile, it uses a deterministic multithreaded mode to improve runtime.

Routing results are repeatable for a given number of cores, so that quality of results will not vary between runs if the same number of cores is specified using the set_host_options command. Results will differ between runs which use different numbers of cores (so for example, two identical runs using 4 cores will produce the same result as each other, and two identical runs using 8 cores will match, but the 4 core runs will differ from the 8 core runs.)

When set to *false*, Zroute global routing will use a single core for routing during compile.

This variable does not affect the behavior of global routing performed as part of the report_congestion command.

Feedback

See Also

- set_host_options
- set_ahfs_options
- report_congestion
- placer_enable_enhanced_router
- placer_congestion_effort

---

spg_enable_via_ladder_opto

Enable via ladder optimization by setting this variable before *compile_ultra -spg*.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Group

syn-qor

Description

Controls the support of via ladder optimization at the end of *compile_ultra -spg*.

The default value for this variable is false.

See Also

- compile_ultra

---

spg_enable_zroute_layer_promotion

Enables the promotion of nets to multiple layers using global routing.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

When enable the variable, Design Compiler Graphical chooses a range of layers for promotion. Candidate nets are chosen based on length, and the tool uses global routing to determine the best set of nets to promote and the exact range of layers for each net. The pattern-based matching can also be used to control the candidate nets.

Setting the variable to *true* uses Zroute for global routing; therefore, it might have a significant impact on runtime.

---

spg_enhanced_timing_model

Enables the use of enhanced timing models for RC estimation when using the default RC extraction.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

Controls the support of enhanced timing models for RC estimation when using the default RC extraction. You do not have to use this variable when using the new RC extraction engine that is enabled with the *spg_icc2_rc_correlation* variable. This is because the new RC extraction engine already uses the advanced RC model to improve the RC correlation with the IC Compiler II tool.

To enable the use of enhanced timing models for RC estimation, set the variable to *true*.

To see the current value of this variable, use the *printvar spg_enhanced_timing_model* command.

See Also

- extract_rc
- spg_icc2_rc_correlation

Feedback

spg_high_effort_mux_area_structuring

Enables high-effort area structuring optimization.

Data Types

Boolean

Default true

Description

This variable enables high-effort dedicated area structuring targeting mux and select tree structures. The optimization allows these structures to share control logic with common controls in order to improve the design area.

spg_icc2_rc_correlation

Performs a new RC extraction to improve the RC correlation with the IC Compiler II tool

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode and Design Compiler NXT topographical mode

Description

When you set this variable to true, the extract_rc command performs RC extraction by using the advanced RC model to improve the RC correlation with the IC Compiler II tool.

When you use the compile_ultra, compile_ultra - incremental, and optimize_netlist commands and explicitly run the *extract_rc* command, the tool issues the *RCEX-302* message to indicate that this feature is enabled for optimization and extraction.

This feature is available only in

- The lower technology when metal layer have significant difference in resistance value

- The Design Compiler NXT tool

To determine the current value of this variable, use the spg_icc2_rc_correlation command

See Also

- extract_rc

Feedback

spg_place_enable_precluster

Enables a placer mode change to reduces QoR variations for small design changes.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, DC Explorer, and Design Compiler NXT non-topographical mode

true in Design Compiler topographical mode and Design Compiler NXT topographical mode

Description

When enabling the variable, Design Compiler Graphical placer to be more aware of logic modules and naturally clumped groups of logic in your netlist. In general, the resulting placement will have improved quality with respect to these structures. The placement of modules and logic clumps will be more predictable from run to run. Small changes in your netlist or your constraints will not perturb where modules are placed. Logic modules are more likely to be properly placed near fixed connections such as ports and macros. Logic modules and logic clumps will generally be more self-cohesive. That is, a module is more likely to be placed in one physical area instead of more than one. Note that some logic modules may be composed of several loosely connected clumps. These loosely connected modules will still be divided if it helps QOR.

ssf_current_version

Show the current version supported by SSF reader and writer.

Data Types

string

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, and DC Explorer

1.1 in Design Compiler NXT non-topographical mode and Design Compiler NXT topographical mode

Description

This variable denotes the current version of SSF file reader and writer in the tool. It is a read-only variable.

To determine the current version of SSF file, use the printvar command.

Feedback

See Also

- ssf_supported_versions
- save_ssf
- printvar

---

ssf_supported_versions

Show the supported versions by SSF reader and writer.

Data Types

string

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, and DC Explorer

1.0 1.1 in Design Compiler NXT non-topographical mode and Design Compiler NXT topographical mode

Description

This variable denotes the supported versions of SSF file reader and writer in the tool. It is a read-only variable.

To determine the supported versions of SSF file, use the printvar command.

See Also

- ssf_current_version
- save_ssf
- printvar

---

suppress_errors

Specifies a list of error codes for which messages are to be suppressed during the current shell session.

Data Types

list

Default ""

Feedback

Description

This variable specifies a list of error codes for which messages are to be suppressed during the current shell session. The default is set to no error messages being suppressed.

Examples

The following example demonstrates the use of the variable.

```
set suppress_errors [list OPT-1406 OPT-1407 CMD-024 UIO-9 UIO-78]
```

---

symbol_library

Specifies the symbol libraries to use during schematic generation.

Data Types

string

Default your_library.sdb in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable specifies the symbol libraries to use during schematic generation. This variable is a list of symbol library names.

---

synlib_abort_wo_dw_license

Stops the *compile* command if the DesignWare license is required to compile the current design, but the DesignWare license is not available.

Data Types

Boolean

Default false

Description

This variable ends the *compile* command when DesignWare license is required to compile the current design, but the DesignWare license is not available.

The datapath generator flow requires the DesignWare license. If the DesignWare license is unavailable, the free DesignWare implementation is used in the design. This could result in a suboptimal circuit.

If *synlib_abort_wo_dw_license* is *true*, the *compile* command quits with an error message. You can also set *synlib_wait_for_design_license* to wait for the licenses if all of the licenses are checked out.

See Also

- synlib_wait_for_design_license

---

synlib_dont_get_license

Specifies a list of synthetic library part licenses that the compiler does not automatically check out.

Data Types

list

Default ""

Description

This variable specifies a list of synthetic library part licenses that the compiler does not automatically check out. By default, the compiler checks out all synthetic library part licenses if there is a possibility that they will be used.

When there is only the possibility of a license being used, add the license to this list so that it will not be automatically checked out.

Exceptions exist in that licenses on this list are checked out, but only when a design is being read that requires these licenses or when you manually request them with the *get_license* command.

To determine what licenses are required to read a design, use the *report_design* command.

To determine the legality of the licenses on the list, use the *set_synlib_dont_get_license* command. This command checks each license on the list and issues a warning message if the license cannot be found in the key file. It then sets the *synlib_dont_get_license* variable.

Feedback

See Also

- get_license
- report_design
- set_synlib_dont_get_license

---

synlib_enable_analyze_dw_power

Record power info of ungrouped DesignWare design.

Data Types

Integer

Default 0

Description

This variable enables recording power related info of ungrouped DesignWare design. The recorded info will be used for *analyze_dw_power* command.

When set to the default value of 0, no recording will happen. The report does not show the estimated power for ungrouped DesignWare designs. When set to 1, recording will be enabled. the *analyze_dw_power* command will show slack and power info for both ungrouped and non-ungrouped DesignWare design.

See Also

- analyze_dw_power

---

synlib_hiis_force_on_cells

Specifies a list of design cells on which the compiler is to force hierarchical incremental implementation selection.

Data Types

list

Default ""

Description

This variable specifies a list of design cells on which the compiler is to force hierarchical incremental implementation selection. Even when implementation is set on the subcomponent of a hierarchical synthetic component, the compiler ignores the

implementation attribute on the subcomponent and performs hierarchical incremental implementation selection on the synthetic component.

The list defined by this variable contains valid cell names in the current design. A simple wildcard pattern can be supported by using the *get_cells* command.

Hierarchical cell names are not supported. For example, to force hierarchical incremental implementation selection on cell U0/U1/MULT, do not place U0/U1/MULT in the *synlib_hiis_force_on_cells* list; put MULT in the list.

The compiler performs forced hierarchical incremental implementation selection only on instantiated DesignWare components that have a single implementation. If you add to the *synlib_hiis_force_on_cells* list the cell name of an operator cell that is inferred in the design, the compiler does not perform forced hierarchical incremental implementation selection on the inferred cell. If there are multiple implementations available for a hierarchical DesignWare component, the compiler does not perform hierarchical incremental implementation selection on the DesignWare cell. To establish single implementation on such a cell, use the *set_implementation* command on the cell or use the *set_dont_use* command on all but one implementation.

Examples

The following example places all cell names that begin with the letter *a* in the *synlib_hiis_force_on_cells* list:

```
prompt> set_app_var synlib_hiis_force_on_cells get_cells "a*"
```

The following example places all cell names in the *synlib_hiis_force_on_cells* list:

```
prompt> set_app_var synlib_hiis_force_on_cells get_cells "*"
```

See Also

- get_cells
- set_dont_use
- set_implementation

synlib_iis_use_netlist

Allows netlists of the DesignWare parts to be used instead of timing models for cost comparison during the incremental implementation selection step.

Data Types

Boolean

Default false

Feedback

Description

This variable allows netlists of the DesignWare parts to be used instead of timing models for cost comparison during the incremental implementation selection step. Timing models are created without considering the design context; netlists are mapped in the context of the design.

When set to *true*, netlists of the DesignWare architectures are used to calculate the timing and area information. When *false*, timing models of the DesignWare architectures are used.

See Also

- compile

---

synlib_preferred_ff_chains

Specifies the preferred flip-flop chains for mapping FF chains in DesignWare Library Clock Domain Crossing (CDC) components.

Data Types

list

Default ""

Description

Specifies a list of metastability-hardened multiple flip-flop cells (chains) that Design Compiler uses when mapping the flip-flop chains in DesignWare Library CDC components, such as DW_sync.

For Design Compiler to consider the flip-flop chain cells during mapping, the technology library cells should meet the following requirements:

1. The technology library should contain the single-bit version of the flip-flop that is used in the flip-flop chain. A special string flag, "ff_chains", should exist on the single flip-flop for Design Compiler to find the chained library cells in the technology library.
2. The technology library cells that contain flip-flop chains should contain the same type of flip-flop. That is, all the flip-flops in the flip-flop chain should be the same.
3. The flip-flop chains in the technology library should have special integer flags, "ff_chain_stage", to indicate how many flip-flops are contained in the flip-flop chain cell.

The technology library vendor that creates the flip-flop chain library cell can define these required attributes in the library source code as "user_defined" attributes.

Feedback

The flip-flop chain library cell can contain any number of single flip-flops. Design Compiler maps the flip-flop chains in DesignWare Library CDC components to the appropriate and preferred flip-flop chain library cells.

For example, if an instance of a DesignWare Library CDC component is configured to use four levels of synchronization and the preferred flip-flop chain library cells have 2, 3, and 4 flip-flops, the priority order is (from high to low):

One 4-flip-flop chain

Two 2-flip-flop chains

Four single flip-flops

Note that there are fewer requirements for single flip-flop metastability-hardened cells to be mapped as specified by the *synlib_preferred_ffs* variable.

See Also

- synlib_preferred_ffs

---

synlib_preferred_ffs

Specifies a list of preferred flip-flop (FF) library cells that DC will map multi-stage synchronizer FFs to within DesignWare Library Clock Domain Crossing (CDC) components.

Data Types

list

Default ""

Description

Specifies a list of single flip-flop (FF), metastability-hardened library cells that DC will use when technology mapping synchronizer FF chains in DesignWare Library CDC components (for example DW_sync).

If there are technology FF library cells that are designed specifically for synchronizers (sometimes referred to as "metastability-hardened" cells), you can set them as the preferred sychronizer FF by adding them to the synlib_preferred_ffs list. When DC maps the synchronizer FF chains within DesignWare Library CDC components, it tries to map them to the preferred FF(s).

Note: After mapping, DC can still size the FF cell. So it is still possible for the FF synchronizer chain used within DesignWare Library CDC components to be mapped to a non-preferred FF library cell.

Feedback

See Also

- synlib_preferred_ff_chains

---

synlib_wait_for_design_license

Specifies a list of authorized synthetic library licenses for which the tool is to wait.

Data Types

list

Default ""

Description

Specifies a list of authorized synthetic library licenses for which the tool is to wait.

By default, the tool checks for all design licenses and checks out one that is available. If none of the licenses are available, the tool terminates the command that requires the license. You can override this default behavior by setting a list of licenses as the value of *synlib_wait_for_design_license*. Then, if no licenses are available, the tool does not terminate the command that requires the license, but waits for the listed licenses to become available.

See Also

- get_license

---

synopsys_program_name

Indicates the name of the program currently running.

Data Types

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

Feedback

synopsys_root

Indicates the root directory from which the application was run.

Data Types

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

synthetic_library

Specifies a list of synthetic libraries to use when compiling.

Data Types

list

Default ""

Description

This variable specifies a list of synthetic libraries to use when compiling.

The *synthetic_library* variable works much like the *target_library* variable does for technology libraries. This variable can be set to be a list of zero or more sldb files that you want to use in the *compile* or *replace_synthetic* commands. When synthetic operators or modules are processed in compile, the operators, bindings, modules, and implementations of the specified library or libraries are used. Synthetic libraries are processed in order. So, if two modules in different libraries have the same name, the module in the first listed library is used.

As with target technology libraries, it is sometimes necessary to include your synthetic library as part of the link_library set. This is especially important when you instantiate synthetic modules.

Because HDL files automatically insert synthetic operators in a netlist, it is important to have a synthetic library defined that supports these operators. For this reason, the standard Synopsys library *standard.sldb* is automatically inserted as the first entry in the

Feedback

synthetic_library variable list. Then if a synthetic library is specified to be an empty list, the insertion of *standard.sldb* provides default definitions.

There is no way to disable the standard synthetic library, but you can disable individual modules or implementations by using the *dont_use* command. To replace a particular implementation, disable it with *dont_use*, and use a replacement from your own synthetic library.

See Also

- compile
- replace_synthetic
- target_library

t

target_library

Specifies the list of technology libraries of components to be used when compiling a design.

Data Types

list

Default your_library.db

Description

This variable specifies the list of technology libraries containing the components to be used when compiling a design. The default value is *your_library.db*. Change this value to reflect your library name.

See Also

- compile

template_naming_style

Generates automatically a unique name when a module is built.

Data Types

string

Feedback

Default %s_%p

Description

This variable automatically generates a unique name when a module is built. This variable is one of three string variables that determine the naming conventions for parameterized modules (templates) built into a design through the *elaborate* command or automatically instantiated from an HDL file. The unique name automatically generated uses the module name, parameter names, and parameter values.

The *template_naming_style* variable determines what character or characters appear between the design name and the parameter name. The string value must contain %s, which represents the name of the original design, and %p, which represents the name and value of the parameter or parameters. You can optionally include any ASCII character or characters, or none, between %s and %p. For example, for a design named *DesignName* that has a parameter *parm1*, the default %s_%p causes the name *DesignName_parm1* to be generated.

Further, %s$%p, %s_*_%p, and %s%p would generate respectively the names *DesignName$parm1*, *DesignName_*_parm1*, and *DesignNameparm1*.

If a design has a noninteger parameter (or if *template_naming_style* = *""*), the following definitions are locked down for these variables:

```
template_naming_style = %s_%p
template_parameter_style = %d
template_separator_style = _
```

See Also

- elaborate
- template_parameter_style
- template_separator_style

---

template_parameter_style

Generates automatically a unique name when a module is built.

Data Types

string

Default %s%d

Description

This variable automatically generates a unique name when a module is built. This variable is one of three string variables that determine the naming conventions for parameterized

Feedback

modules (templates) built into a design through the *elaborate* command or automatically instantiated from an HDL file. The unique name automatically generated uses the module name, parameter names, and parameter values.

The *template_parameter_style* variable determines what character or characters appear between the parameter name and its value.

The string must contain %s, which represents the name of the parameter, and %d, which represents the value of the parameter. You can optionally include any ASCII character or characters, or none, between %s and %d. For example, for a parameter named *parm* that has a value of *1*, the default %s%d causes the name *parm1* to be generated.

The following are additional examples:

```
template_naming_style = "%s$%p"
template_parameter_style = %s_%d    results in DesignName$parm_1

template_naming_style = "%s_%p"
template_parameter_style = %s@%d    results in DesignName_parm@1
```

If a design has a noninteger parameter (or if *template_naming_style* = *""*), the following definitions are locked down for these variables:

```
template_naming_style = %s_%p
template_parameter_style = %d
template_separator_style = _
```

See Also

- elaborate
- template_naming_style
- template_separator_style

---

template_separator_style

Generates automatically a unique name when a module is built.

Data Types

string

Default _

Description

This variable automatically generates a unique name when a module is built. This variable is one of three string variables that determine the naming conventions for parameterized modules (templates) built into a design through the *elaborate* command or automatically

Feedback

instantiated from an HDL file. The unique name automatically generated uses the module name, parameter names, and parameter values.

The *template_separator_style* variable determines what character or characters appear between parameter names for templates that have more than one parameter. You can designate any ASCII character or characters, or none. The default value is an underscore (_).

For example, for a design called *DesignName* that has parameters named *parm1*, *parm2*, and *parm3*, if *template_naming_style = "%s_%p"* (the default value), and *template_separator_style = "_"*, the name *DesignName_parm1_parm2_parm3* is generated.

The following are additional examples:

```
template_naming_style = "%s$%p"
template_separator_style = "_"      results
 in DesignName$parm1_parm2_parm3
template_naming_style = "%s#%p"
template_separator_style = "/"      results
 in DesignName#parm1/parm2/parm3
```

If a design has a noninteger parameter (or if *template_naming_style = ""*), the following definitions are locked down for these variables:

```
template_naming_style = %s_%p
template_parameter_style = %d
template_separator_style = _
```

See Also

- elaborate
- template_naming_style
- template_parameter_style

---

test_allow_internal_pins_in_hierarchical_flow

Allows DFT cores with internal pins to be integrated in a hierarchical DFT flow.

Data Types

Boolean

Default false

Feedback

Description

By default, cores with internal pins cannot be integrated, for the safety reasons described in the TESTXG-68 man page.

If your flow requires such cores to be integrated and you can guarantee proper operation of the cores during hierarchical testing, then you can set this variable to *true* to allow these cores to be integrated.

The variable must be set to *true* as follows:

- During core-level creation, to include additional internal-pins information in the core's CTL model
- During top-level insertion, to disable the TESTXG-68 safety check and allow the cores to be integrated

Cores with internal clock pins (other than OCC *oscillator* pins) are not supported by this variable because they cannot be effectively represented in the CTL model.

See Also

- dft_drc

---

test_ate_sync_cycles

Specifies the number of ATE clock synchronization cycles used by a user-defined on-chip clocking (OCC) controller.

Data Types

positive integer

Default 1

Description

When you are using a user-defined on-chip clocking (OCC) controller, this variable specifies the number of ATE clock synchronization cycles used by the OCC controller when a scan-enable transition initiates a transition between the ATE and OCC clocks.

This variable affects the content of the protocol file written at the end of the *insert_dft* command.

By default, this variable is set to 1, which models an OCC controller that waits for a single clock cycle after a scan-enable transition. When scan-enable is reasserted after capture, scan data is delayed by one cycle, and the load_unload procedure is created as follows:

```
load_unload {
  V { SE=1; ATE_CLK=P; _si-=X; _so=X}
```

```
  Shift { ...}
}
```

You should change the value of this variable if your OCC controller does not need a synchronization cycle or if it requires more than one synchronization cycle after a scan-enable transition.

For more information on the synchronization logic structures and timing relationships, see SolvNet article 035708, "What Does the test_ate_sync_cycles Variable Do?".

Normally, this variable only affects the test protocol. If the variable is not set to the proper value, the test protocol can be edited or regenerated. However, in serialized compressed scan flows, it also affects the architecture of the serializer clock controller. In these flows, if the variable is not set to the proper value, the serializer logic must be reinserted with the variable set to the correct value.

See Also

- set_dft_configuration
- set_scan_compression_configuration
- set_dft_clock_controller

---

test_avoid_control_register_of_icg_in_scan_chain_head

Forces all control registers of clock-gating functional enable pins to be at the end of a scan chain.

Data Types

Boolean

Default false

Description

By default, the tool has no restrictions on the scan chain location of scan registers that drive the functional enable pins of clock-gating cells. If these control registers are at the head (beginning) of a scan chain, then the design can lose test coverage.

If you set this variable to *true*, the tool places the registers at the ends of scan chains, and it forces the SCANDEF information to have the STOP pin be at the control register to prevent it from moving earlier in the chain.

This variable only affects control registers that are in the same hierarchy as the clock-gating cell.

Feedback

See Also

- set_scan_path
- identify_clock_gating

---

test_bsd_dead_cycle_after_update_dr

Adds dead cycles before EXTEST measurement cycle so that slow output pads can be measured.

Data Types

integer

Default 0 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

For designs with slow I/O pads or fast TCK clock frequencies or both, when the EXTEST instruction is active, the output pads might not settle before the value is measured in the clock cycle after the Update-DR state. In this case, you can add one or more dead cycles after the Update-DR state when the EXTEST instruction is active, which provide additional settling time for slow output pads.

You can use this variable to set the desired number of dead cycles. For example,

```
prompt> set_app_var test_bsd_dead_cycle_after_update_dr cycle_count
```

When the *create_bsd_patterns* command is run, it creates patterns with the specified number of dead cycles before output measurement. TCK will not pulse during these dead cycles. The default is zero, which does not insert any dead cycles.

See Also

- create_bsd_patterns

---

test_bsd_default_bidir_delay

Defines the default switching time of bidirectional ports in a tester cycle for BSD applications.

Data Types

positive nonzero integer

Feedback

Default 0 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable defines the default switching time of bidirectional ports in a tester cycle. The value is a positive real number in nanoseconds. This value is used when a test protocol is generated for ATPG.

For ATPG, *test_bsd_default_bidir_delay* defines the following for the design under test:

- The default time at which values are applied (driven) to the bidirectional ports in input mode during the parallel measure cycle.
- The time at which bidirectional ports are released (undriven) during the capture cycle.
- This variable affects only BSD applications.

The value of *test_bsd_default_bidir_delay* must be less than the output strobe time and greater than the capture clock active edge value.

The value of this variable affects the outcome of the *check_bsd* command and the test program produced by the *create_bsd_patterns* or *write_test* commands. The *check_bsd* command uses the value of this variable to check a design against the design rules of a bsd methodology.

Changing the value of *test_bsd_default_bidir_delay* modifies the content of the inferred or created test protocol generated by *check_bsd*. To generate the test program, *create_bsd_patterns* or *write_test* uses the value of *test_bsd_default_bidir_delay* in the test protocol attached to the vector database (.vdb) file.

See Also

- check_bsd
- write_test
- port_attributes
- test_bsd_default_delay
- test_bsd_default_strobe
- test_bsd_default_strobe_width

Feedback

test_bsd_default_delay

Defines the default time in a tester cycle to apply values to input ports for BSD applications.

Data Types

positive nonzero integer

Default 0 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable defines the default time in a tester cycle to apply values to input ports. The value is a positive real number in nanoseconds. This variable is used to generate a protocol file for ATPG.

For ATPG, *test_bsd_default_delay* defines for the design under test the default time at which values are applied (driven) to the primary inputs for BSD applications. The value of *test_bsd_default_delay* must be less than the output strobe time, and less than the capture clock edge value.

The value of this variable affects the outcome of the *check_bsd* command and the test program produced by the *create_bsd_patterns* or *write_test* commands. The *check_bsd* command uses the value of this variable to check a design against the design rules of a bsd methodology. Changing the value of *test_bsd_default_delay* modifies the content of the inferred or created BSD protocol generated by *check_bsd*.

To generate the test program, *create_bsd_patterns* or *write_test* uses the value of *test_bsd_default_delay* in the test protocol attached to the vector database (.vdb) file.

See Also

- check_bsd
- write_test
- port_attributes
- test_bsd_default_bidir_delay
- test_bsd_default_strobe
- test_bsd_default_strobe_width

Feedback

test_bsd_default_strobe

Defines the default strobe time in a test cycle for output ports and bidirectional ports in output mode for BSD applications.

Data Types

positive nonzero integer

Default 95 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable defines the default strobe time in a test cycle for output ports and bidirectional ports in output mode for BSD applications. The value is a positive real number in nanoseconds. This value is used when a test protocol is generated for ATPG.

For ATPG, *test_bsd_default_strobe* defines for the design under test the default time at which values are strobed on the primary output ports and bidirectional ports in output mode. The value of this variable must be less than or equal to the clock period value.

The value of this variable affects the outcome of the *check_bsd* command and the test program produced by the *create_bsd_patterns* or *write_test* command. The *check_bsd* command uses the value of this variable to check a design against the design rules of a bsd methodology.

Changing the value of *test_bsd_default_strobe* modifies the content of the inferred or created test protocol generated by *check_bsd*. To generate the test program, *create_bsd_patterns* or *write_test* uses the value of *test_bsd_default_strobe* in the test protocol attached to the vector database (.vdb) file.

See Also

- check_bsd
- write_test
- test_bsd_default_bidir_delay
- test_bsd_default_delay
- test_bsd_default_strobe_width

Feedback

test_bsd_default_strobe_width

Defines the default strobe pulse width, which is the default time that specifies how long after invocation the strobe pulse needs to be held active for BSD applications.

Data Types

positive nonzero integer

Default 0 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable defines the default strobe pulse width, which is the default time that specifies how long after invocation the strobe pulse needs to be held active for BSD applications.

Changing the value of this variable modifies the content of the inferred or created test protocol generated by the *check_bsd* command. The *create_bsd_patterns* or *write_test* command uses the value of this variable (as stated in the test protocol attached to the design) to generate the test program for vector formats that support the notion of strobe width. For those formats that do not support it, the strobe width value specified is ignored.

The value of the *test_bsd_default_strobe_width* variable is a real number value whose units are assumed to be nanoseconds, and the value must be a positive number. The sum of this value and the strobe time value must be less than or equal to the clock period value.

See Also

- check_bsd
- write_test
- test_bsd_default_bidir_delay
- test_bsd_default_delay
- test_bsd_default_strobe

test_bsd_input_ac_parametrics

Specifies what input port value transitions are tested in DC parametric patterns.

Feedback

Data Types

Boolean

Default false

Description

By default, the DC parametric patterns created by the *create_test_patterns -type {dc_parametric}* command exercise all port values for leakage characterization, but they do not exercise all value transitions for switching characterization.

With the default of *false*, input ports and bidirectional ports in input mode go through the following value sequence:

```
Input ports:
  X -> 0
  0 -> 1

Bidirectional ports as inputs:
  Z -> 0
  0 -> 1
```

By setting this variable to *true*, input ports and bidirectional ports input mode go through the following value sequence:

```
Input ports:
  X -> 0
  0 -> 1
  1 -> 0

Bidirectional ports as inputs:
  Z -> 0
  0 -> 1
  1 -> 0
```

See Also

- test_bsd_new_output_parametrics

---

test_bsd_make_private_instructions_public

Specifies whether private instructions should be included in BSDL and pattern files.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable controls whether private instructions are included during BSDL and pattern file generation. Private instructions are created using the *-private* option of the *set_bsd_instruction* command.

When this variable is set to *false* (the default), the *write_bsdl* and *create_bsd_patterns* commands ignore private instructions and their associated test data registers. This behavior is useful for creating BSDL and pattern files for external customers.

When this variable is set to *true*, the *write_bsdl* and *create_bsd_patterns* commands include private instructions. The *write_bsdl* command includes private instructions and any associated test data registers in the REGISTER_ACCESS section of the BSDL output file, and the *create_bsd_patterns* command includes private instructions in test pattern generation. This behavior is useful for creating BSDL and pattern files for internal testing.

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

To see the current value of this variable, type

prompt> *printvar test_bsd_make_private_instructions_public*

Examples

The following example generates BSDL and pattern files for external use:

```
prompt> set test_bsd_make_private_instructions_public false

prompt> write_bsdl -naming_check BSDL -output des.bsdl

prompt> create_bsd_patterns -type all -output patterns.spf
```

The following example generates BSDL and pattern files for internal testing:

```
prompt> set test_bsd_make_private_instructions_public true

prompt> write_bsdl -naming_check BSDL -output des_INTERNAL.bsdl

prompt> create_bsd_patterns -type all -output patterns_INTERNAL.spf
```

To determine the current value of this variable, type *printvar test_bsd_make_private_instructions_public*. For a list of *bsd* variables and their current values, type *print_variable_group bsd*.

Feedback

See Also

- set_bsd_instruction
- write_bsdl
- create_bsd_patterns

test_bsd_manufacturer_id

Specifies the manufacturer ID to use to create the value captured in the device identification register during execution of the *insert_dft* command.

Data Types

integer

Default 72 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable specifies the manufacturer ID to use to create the value captured in the device identification register during execution of the *insert_dft* command. This unique ID represents the manufacturer ID assigned to the organization.

See Also

- insert_dft
- test_bsd_part_number
- test_bsd_version_number

test_bsd_new_output_parametrics

Specifies what output port value transitions are tested in DC parametric patterns.

Data Types

Boolean

Default false

Feedback

Description

By default, the DC parametric patterns created by the *create_test_patterns -type {dc_parametric}* command exercise all port values for leakage characterization, but they do not exercise all value transitions for switching characterization.

With the default of *false*, two-state output ports, three-state output ports, and bidirectional ports in output mode go through the following value sequence:

```
Two-state output ports:
  X -> 0
  0 -> 1

Three-state output ports, bidirectional ports as outputs:
  Z -> 0
  0 -> 1
```

By setting this variable to *true*, two-state output ports, three-state output ports, and bidirectional ports in output mode go through the following value sequence:

```
Output ports:
  X -> 0
  0 -> 1
  1 -> 0
  X -> 1

Three-state output ports, bidirectional ports as outputs:
  Z -> 0
  0 -> 1
  1 -> 0
  Z -> 1
```

See Also

- test_bsd_input_ac_parametrics

---

test_bsd_optimize_control_cell

Allows the *optimize_bsd* command to optimize allocation of BSR control cells during area-driven optimization.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Feedback

Description

This variable allows the *optimize_bsd* command to optimize allocation of boundary-scan register (BSR) control cells during area-driven optimization.

using the value of the *test_bsd_control_cell_drive_limit* variable. If the area constraints are not violated, this optimization is not performed. To specify area constraints for the current design, use the *set_max_area* command.

When set to *false* (the default), no optimization of BSR control cells takes place during area-driven optimization.

When set to *true*, optimization is performed if the *set_max_area* constraint for the current design is violated. In this case, optimization is performed using the value of the *test_bsd_control_cell_drive_limit* variable. If the area constraints are not violated, this optimization is not performed.

To determine the current value of this variable, type *printvar test_bsd_optimize_control_cell*. For a list of *bsd* variables and their current values, type *print_variable_group bsd*.

See Also

- set_max_area

---

test_bsd_part_number

Specifies the part number to use to create the value captured in the device identification register during execution of the *insert_dft* command.

Data Types

integer

Default 72 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable specifies the part number to use to create the value captured in the device identification register during execution of the *insert_dft* command. This number, which represents the part number assigned to the IC, should be unique inside the organization.

Feedback

See Also

- insert_dft
- test_bsd_version_number
- test_bsd_manufacturer_id

---

test_bsd_synthesis_gated_tck

Specifies whether BSD insertion gates the clock connection of a user-defined test data register with its instruction enable signal.

Data Types

Boolean

Default false in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable specifies whether BSD insertion gates the TCK clock connection of a user-defined test data register (UTDR) with its instruction enable signal.

When this variable is set to *false*, which is the default, BSD insertion hooks the UTDR clock pin directly up to TCK. Any existing connections are discarded. No gating logic is inserted to suppress TCK when the instruction is inactive.

When this variable is set to *true*, BSD insertion gates the TCK signal with a MUX, with the select pin driven by the instruction enable signal. The MUX selects TCK when the instruction is active, and it selects the existing connection when the instruction is inactive.

Examples

Consider the following UTDR definition:

```
set_dft_signal -view spec -type tdi \\
  -hookup_pin BIST/WRAPPER_0/debug_in
set_dft_signal -view spec -type tdo \\
  -hookup_pin BIST/WRAPPER_0/debug_out
set_dft_signal -view spec -type bsd_shift_en \\
  -hookup_pin BIST/WRAPPER_0/debug_en
set_dft_signal -view spec -type capture_clk \\
  -hookup_pin BIST/WRAPPER_0/clk

set_scan_path DEBUG_REG -class bsd \\
    -view spec \\
```

Feedback

```
-hookup {BIST/WRAPPER_0/debug_in \\
    BIST/WRAPPER_0/debug_out \\
    BIST/WRAPPER_0/debug_en \\
    BIST/WRAPPER_0/clk} \\
-exact_length 10
```

When this variable is set to *false*, which is the default, BSD insertion hooks the UTDR clock pin BIST/WRAPPER_0/clk directly to TCK, and the existing connection is lost.

When this variable is set to *true*, BSD insertion MUXes the existing signal at the UTDR clock pin BIST/WRAPPER_0/clk with TCK, using the instruction enable signal as the MUX select, so that TCK only drives the clock pin when the instruction is active.

To determine the current value of this variable, type *printvar test_bsd_synthesis_gated_tck*. For a list of *bsd* variables and their current values, type *print_variable_group bsd*.

See Also

- set_bsd_instruction
- set_scan_path

---

test_bsd_version_number

Specifies the version number to use to create the value captured in the device identification register during execution of the *insert_dft* command.

Data Types

integer

Default 72 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable specifies the version number to use to create the value captured in the device identification register during execution of the *insert_dft* command. This represents the version number assigned to this IC and is unique for this IC.

Feedback

See Also

- insert_dft
- test_bsd_manufacturer_id
- test_bsd_part_number

---

test_bsdl_default_suffix_name

Specifies the default suffix for the name of the BSDL file generated by the *write_bsdl* command.

Data Types

string

Default bsdl in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable specifies the default suffix for the name of the Boundary-Scan Description Language (BSDL) file that the *write_bsdl* command generates. If the *write_bsdl* command does not specify an output file, by default the output file is named *top_level_design_name.suffix*, with *suffix* being the value of *test_bsdl_default_suffix_name*.

See Also

- write_bsdl

---

test_bsdl_max_line_length

Specifies the maximum number of characters per line for the output BSDL file the *write_bsdl* command produces.

Data Types

integer

Default 95 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Feedback

Description

This variable specifies the maximum number of characters per line for the output Boundary-Scan Description Language (BSDL) file the *write_bsdl* command produces. The maximum line length should not be less than 50 or more than 132. If you specify a line length outside this range, *write_bsdl* uses the default instead.

See Also

- write_bsdl

---

test_cc_ir_masked_bits

Identifies instruction register (IR) bits to be masked during the search by the *check_bsd* command for all possible implemented instructions.

Data Types

integer

Default 0

Description

This variable identifies instruction register (IR) bits to be masked during the search by the *check_bsd* command for all possible implemented instructions. By default, no bits are masked.

Masking IR bits shortens the sequential search. The sequential search becomes exponentially costly as the IR length increases beyond 8 bits. The compliance checker masks bits that contain a 1 in the binary equivalent of the decimal integer in this variable. For example, if you want to mask bits 0 and 3, set the variable with decimal 9, the equivalent of binary 00...1001. Similarly, a value of 7 (binary 00...0111) masks bits 0, 1 and 2; 8 masks bit 3; and so on.

See Also

- check_bsd
- test_cc_ir_value_of_masked_bits

---

test_cc_ir_value_of_masked_bits

Specifies values to be forced into bits of the instruction register (IR) that are masked, during the search by the *check_bsd* command for all possible implemented instructions.

Feedback

Data Types

integer

Default 0

Description

This variable specifies values to be forced into bits of the instruction register (IR) that are masked, during the search by the *check_bsd* command for all possible implemented instructions. The value of the *test_cc_ir_masked_bits* variable identifies the bits to be masked. By default, *0*s are forced into all masked bits.

The compliance checker forces a 0 or a 1 into each masked bit, based on the binary equivalent of the decimal integer value of *test_cc_ir_value_of_masked_bits*, adjusted for the bit-width of the IR. For example, for a 4-bit IR, if bits 0 and 3 are masked and the value of *test_cc_ir_value_of_masked_bits* is 5 (binary 0101), bit 0 receives a value of 1 and bit 3 a value of 0.

See Also

- check_bsd
- test_cc_ir_masked_bits

---

test_clock_port_naming_style

Specifies the naming style used by the *insert_dft* command for global test signal ports created in designs during the addition of test circuitry.

Data Types

string

Default test_c%s

Description

This variable specifies the naming style used by the *insert_dft* command for global test signal ports created in designs during the addition of test circuitry. If the *set_dft_signal* command identifies suitable ports, new ports are not added.

This variable must contain one %s (percent s) character sequence. When adding a port, if the new port name is the same as an existing port name, %s is replaced by a character (*a* through *z*) that creates a unique name.

For example, if this variable is set to *MY_TC%s* when *insert_dft* adds a new port for the *test clock* signal, the port is named:

MY_TC

If a port named *MY_TC* already exists in the design, the new port is named:

MY_TCa

See Also

- insert_dft
- set_dft_signal

---

test_core_wrap_sync_ctl_segment_length

Specifies the maximum length of CTL-modeled synchronizer register that can be included in wrapper chains.

Data Types

integer

Default 0

Description

This variable specifies the maximum segment length of synchronizer register that can be included in wrapper chains as shared wrapper cells in the maximized reuse core wrapping flow. If a synchronizer register longer than this is associated with a port, that port gets a dedicated wrapper cell.

A synchronizer register is a library cell register that has a CTL model with a single scan chain from scan data input to scan data output. Note that a multibit register is something different, and is not affected by this variable.

The default is 0, which indicates that no CTL-modeled register (including single-bit registers) can be included in wrapper chains. Note that in this default case, when tool finds any CTL-modeled cell associated with a design port, a dedicated wrapper cell is added to the port.

By default, synchronizer registers always count as 1 in the fanout threshold computation, regardless of internal synchronizer length. To make them count according to their segment length, set the *test_core_wrap_use_sync_ctl_segment_length_for_fanout_check* variable to *true*.

If a CTL-modeled cell other than a synchronizer register is associated with a port, that port always gets a dedicated wrapper cell.

This variable is used only in maximized reuse core wrapper flows.

This variable does not apply to synchronizer registers modeled using Liberty library constructs.

Feedback

Examples

To allow typical dual-register synchronization cells to be included in wrapper chains,

```
prompt> set_app_var test_core_wrap_sync_ctl_segment_length 2
```

See Also

- test_core_wrap_use_sync_ctl_segment_length_for_fanout_check

---

test_core_wrap_use_sync_ctl_segment_length_for_fanout_check

Specifies whether maximized-reuse fanout computations use the segment length for CTL-modeled cells.

Data Types

Boolean

Default false

Description

This variable specifies how the maximized reuse fanout computation should treat the segment length of CTL-modeled synchronizer registers.

A synchronizer register is a library cell register that has a CTL model with a single scan chain from scan data input to scan data output. (Note that a multibit register is something different, and is not affected by this variable.)

When set to *false* (the default), synchronizer registers always count for 1 in the threshold computation, regardless of their segment length. This reflects fanin/fanout count for a port.

When set to *true*, synchronizer registers count as their segment length in the threshold computation. This reflects total scan element count associated with a port.

This variable is used only in maximized reuse core wrapper flows.

This variable does not apply to synchronizer registers modeled using Liberty library constructs.

See Also

- test_core_wrap_sync_ctl_segment_length

---

test_count_occ_chain_in_chain_count

Controls whether external OCC chains are included in the chain count constraint.

Feedback

Data Types

Boolean

Default false

Description

This variable affects only designs that have DFTMAX Ultra compression modes.

When this variable is set to *false* (the default), external clock chains are not included in the *-chain_count* specification of the *set_scan_configuration* command, which means that the actual number of chains architected (including clock chains) is larger than the chain count specification.

This behavior is different than other test modes (compressed and uncompressed), and it is different than the behavior of non-OCC external chains, which can lead to confusion. Although the behavior only occurs when DFTMAX Ultra compression is used, it is applied to other test modes as a result.

When this variable is set to *true*, the chain count is included in the *-chain_count* specification of the *set_scan_configuration* command. This behavior is consistent with other test modes and with the behavior of non-OCC external chains.

See Also

- set_scan_configuration

---

test_debug_print_wrp_excluded_ports

Allows user to print out a report of all excluded ports from wrapper classified into 2 categories 1. excluded by user 2. excluded by tool

Data Types

String

Default ""

Description

This variable allows user to print out a report of all excluded ports from wrapper classified into 2 categories 1. excluded by user 2. excluded by tool.

Examples

For example,

```
set test_debug_print_wrp_excluded_ports excluded_ports.rpt
```

Feedback

See Also

- insert_dft

---

test_dedicated_clock_chain_clock

Enables a dedicated clock-chain clock output for DFT-inserted OCC controllers.

Data Types

Boolean

Default false

Description

By default, when the tool inserts an on-chip clocking (OCC) controller and its clock chain, the clock chain clock connection shares the first functional clock output of the OCC controller. This places the clock chain in both the PLL and ATE clock paths.

To use a dedicated clock-chain clock connection from the OCC controller design, set this variable to *true*. This creates a dedicated OCC controller clock output for the clock chain that places it in only the ATE clock path, which prevents it from affecting the high-speed PLL clock path.

In both cases, you should consider how the clock-chain connection interacts with clock tree synthesis (CTS). Depending on the value of this variable and the CTS configuration used for the PLL/ATE clocks, the clock chain might or might not be included in a synthesized clock clock tree. If you are performing CTS at the functional OCC controller output but you are using a dedicated clock chain clock connection, consider hold time effects within the clock chain (if no CTS is performed for the ATE clock domain) and shifting to/from the clock chain (if the clock latency to the clock chain differs from that of the functional clock domain).

This variable can be set to *true* only when the *test_occ_insert_clock_gating_cells* variable is also set to *true*.

See Also

- set_dft_clock_controller
- test_occ_insert_clock_gating_cells

---

test_dedicated_subdesign_scan_outs

Instructs the tool to create dedicated scan-out ports on subdesigns.

Feedback

Data Types

Boolean

Default false

Description

This variable instructs the tool to always create dedicated scan-out ports on subdesigns. By default, the tool uses existing subdesign output ports where possible. When this variable is set to *true*, the tool creates new dedicated output ports for scan.

This variable only affects cases where the Q output of flop is directly connected to output of a subblock. If the variable is set to *true*, the tool creates a new, separate output port driven by the register. As a result, the register output net becomes a multiport net and buffers must be added to fix the multiport condition.

If the variable is set to *false* (the default), the functional output port driven by the register gets reused as a scan-out.

If there is no register that directly drives an output port, the tool does not reuse a functional port via a MUX; instead, it creates a new scan-out port.

See Also

- insert_dft

---

test_default_bidir_delay

Defines the default switching time of bidirectional ports in a tester cycle.

Data Types

positive integer

Default 0

Description

This variable defines the default switching time of bidirectional ports in a tester cycle. The value is a positive real number in nanoseconds. This value is used when a test protocol is generated for ATPG.

For ATPG, *test_default_bidir_delay* defines for the design under test the following:

```
• The default time at which values are applied (driven)
to the bidirectional ports in input mode during
the parallel measure cycle.
```

Feedback

```
• The time at which bidirectional ports are released
(undriven) during the capture cycle.
```

The value of *test_default_bidir_delay* must be less than the output strobe time and less than the capture clock active edge value.

The value of this variable affects the outcome of the *dft_drc* command and the test protocol produced by the *write_test_protocol* command. The *dft_drc* command uses the value of this variable to check a design against the design rules of a scan test methodology.

Changing the value of *test_default_bidir_delay* modifies the content of the inferred or created test protocol generated by *dft_drc*. You can override the value of this variable by reading in a user-defined test protocol using the *read_test_protocol* command.

See Also

- dft_drc
- read_test_protocol
- write_test
- write_test_protocol
- port_attributes
- test_default_delay
- test_default_period
- test_default_strobe
- test_default_strobe_width

---

test_default_delay

Defines the default time in a tester cycle to apply values to input ports.

Data Types

positive integer

Default 0

Description

This variable defines the default time in a tester cycle to apply values to input ports. The value is a positive real number in nanoseconds. The default value is 0.0. This variable is used to generate a protocol file for ATPG.

Feedback

For ATPG, *test_default_delay* defines for the design under test the default time at which values are applied (driven) to the primary inputs. The value of *test_default_delay* must be less than the output strobe time, and less than the capture clock edge value.

The value of this variable affects the outcome of the *dft_drc* command and the test program produced by the *write_test* command. The *dft_drc* command uses the value of this variable to check a design against the design rules of a scan test methodology.

Changing the value of *test_default_delay* modifies the content of the inferred or created test protocol generated by *dft_drc* or *create_test_protocol*. You can override the value of this variable by reading in a user-defined test protocol using the *read_test_protocol* command. To generate the test program, *write_test* uses the value of *test_default_delay* in the test protocol attached to the vector data base (.vdb) file.

See Also

- dft_drc
- read_test_protocol
- write_test
- write_test_protocol
- port_attributes
- test_default_bidir_delay
- test_default_period
- test_default_strobe
- test_default_strobe_width

---

test_default_period

Defines the length of a test vector cycle.

Data Types

positive nonzero integer

Default 100

Description

This variable defines the time duration of a test vector cycle. This value translates directly to the speed of application of the test vectors on automated test equipment (ATE). For example, a period of 100 nanoseconds describes ATE running at 10 MHz.

Feedback

This variable controls the test clock period of the test protocol created by the *create_test_protocol* command. Test clocks defined with the *set_dft_signal* command must be defined using this period, except for free-running reference clocks defined with *set_dft_signal -type refclock*, which can be defined with any period value.

If you read in an external test protocol with the *read_test_protocol* command, the test period used in that protocol file is used, and the value of this variable is ignored.

The *create_bsd_patterns* command uses the value of this variable when creating the boundary-scan test program.

The value of the *test_default_period* variable is a real number whose units are assumed to be nanoseconds. The value must be a positive number.

See Also

- dft_drc
- read_test_protocol
- set_dft_signal
- write_test
- write_test_protocol
- test_default_bidir_delay
- test_default_delay
- test_default_strobe
- test_default_strobe_width
- test_wrapper_new_wrp_clock_timing

---

test_default_strobe

Defines the default strobe time in a test cycle for output ports and bidirectional ports in output mode.

Data Types

positive nonzero integer

Default 40

Feedback

Description

This variable defines the default strobe time in a test cycle for output ports and bidirectional ports in output mode. The value is a positive real number in nanoseconds. This value is used when a test protocol is generated for ATPG.

For ATPG, *test_default_strobe* defines, for the design under test, the default time at which values are strobed on the primary output ports and bidirectional ports in output mode. The value of this variable must be less than or equal to the clock period value.

The value of this variable affects the outcome of the *dft_drc* command and the test program produced by the *write_test* command. The *dft_drc* command uses the value of this variable to check a design against the design rules of a scan test methodology.

Changing the value of *test_default_strobe* modifies the content of the inferred or created test protocol generated by *dft_drc*. You can override the value of this variable for by reading in a user-defined test protocol using the *read_test_protocol* command. To generate the test program, *write_test* uses the value of *test_default_strobe* in the test protocol attached to the vector database (.vdb) file.

See Also

- dft_drc
- read_test_protocol
- write_test
- write_test_protocol
- test_default_bidir_delay
- test_default_delay
- test_default_period
- test_default_strobe_width

---

test_default_strobe_width

Defines the default strobe pulse width, which is the default time that specifies how long after invocation the strobe pulse needs to be held active.

Data Types

positive integer

Default 0

Feedback

Description

This variable defines the default strobe pulse width, which is the default time that specifies how long after invocation the strobe pulse needs to be held active.

Changing the value of this variable modifies the content of the inferred or created test protocol generated by the *dft_drc* command. You can overwrite the value of this variable by providing a user-defined test protocol and reading it in using the *read_test_protocol* command. The *write_test* command uses the value of this variable (as stated in the test protocol attached to the design) to generate the test program for vector formats that support the notion of strobe width. For those formats that do not support it, the strobe width value specified is ignored.

The value of the *test_default_strobe_width* variable is a real number value whose units are assumed to be nanoseconds, and the value must be a positive number. The sum of this value and the strobe time value must be less than or equal to the clock period value.

See Also

- dft_drc
- read_test_protocol
- write_test
- write_test_protocol
- test_default_bidir_delay
- test_default_delay
- test_default_period
- test_default_strobe

---

test_disable_enhanced_dft_drc_reporting

Prevents the *dft_drc* command from reporting DRC violations using a new format.

Data Types

Boolean

Default true

Description

This variable prevents the *dft_drc* command from reporting DRC violations using a new format. The default is *true*, which disables the new format and uses the old DRC format for backward compatibility. To use the new format, set this variable to *false*.

Feedback

See Also

- dft_drc

---

test_disable_find_best_scan_out

Selects the scan-out pin on a scan cell based on availability instead of timing slack.

Data Types

Boolean

Default false

Description

This variable, when set to *false* causes the *insert_dft* command to select the scan-out pin on a scan cell that has the greatest timing slack. This usually means that *insert_dft* makes extensive use of Qbar pins to drive the scan-in pins of scan cells. The tool supports inversions on the scan chain.

If you are using tools that do not support inversions, you can disable this behavior by setting this variable to *true*. The *insert_dft* command does not attempt to find the optimum driver. Instead, the tool selects the driver based on availability, and it chooses the dedicated scan-out driver if any are defined in the library. The chosen driver can still be inverting or noninverting.

See Also

- insert_dft

---

test_dont_fix_constraint_violations

Minimizes performance constraint violations.

Data Types

Boolean

Default false

Description

This variable minimizes performance constraint violations. When the value of this variable is set to *false* (the default), the *insert_dft* command attempts to minimize performance constraint violations. You can disable this behavior by setting the variable to *true*.

Feedback

See Also

- insert_dft

---

test_enable_capture_checks

Controls checking for capture violations during execution of the *dft_drc* command.

Data Types

Boolean

Default true

Description

This variable controls the checking for capture violations, during the execution of the *dft_drc* command.

Setting the value of *test_enable_capture_checks* to *true* (the default value) enables capture checks during *dft_drc* command activity.

Setting the value of *test_enable_capture_checks* to *false* disables capture checks during *dft_drc* command activity.

See Also

- dft_drc

---

test_enable_codec_sharing

Instructs DFTMAX to share the same CODEC architecture across all partition decompressors/compressors.

Data Types

Boolean

Default false

Description

When you insert adaptive scan compression logic with DFTMAX for multiple partitions at the top level, this variable controls whether a shared CODEC architecture is used across all partition decompressors/compressors (CODECs).

When set to its default value of *false*, smaller partition-specific CODECs are created for each partition. The top-level scan-in and scan-out ports are allocated separately to each partition CODEC, according to the total scan chain counts defined for the Internal_scan

Feedback

and ScanCompression_mode modes for each partition. Each scan-in and scan-out port connects to a single partition CODEC. There must be a sufficient number of scan-in and scan-out ports available to satisfy all partition CODEC connections.

When you set this variable to *true*, the scan chain counts are summed across the Internal_scan and ScanCompression_mode modes to build a more adaptive CODEC that is shared by all partitions. However, instead of building one large CODEC that connects to all partitions, this common CODEC is replicated for each partition. Functionally, all partitions share the same CODEC. Architecturally, this common CODEC is duplicated to keep routing local within the partition and to minimize top-level congestion.

For each duplicated CODEC, a separate subset of the decompressor outputs and compressor inputs are hooked up to each partition according to the ScanCompression_mode scan chain count for that partition. The scan-in ports are shared, so that each scan-in port drives all CODECs. The scan-out ports are combined using XOR logic so that the top-level scan-out ports can capture the output from all partition compressors scan outputs. This architecture has similar testability efficiency to having a single top-level CODEC with the same architecture.

With the shared CODEC architecture, each partition's CODEC has only its subset of compressor and decompressor signals connected. After running *insert_dft*, you should perform an incremental compile with boundary scan optimization enabled to optimize the CODEC logic to each partition.

See Also

- set_dft_configuration
- set_scan_compression_configuration
- define_dft_partition

---

test_enable_retiming_flops_driven_by_direct_scan_clock_driver

Controls whether retiming registers are clocked by the global or locally gated clock.

Data Types

Boolean

Default false

Group

test_variables

Feedback

Description

By default, the clock pin of a retiming register is connected to the clock source signal, even when the scan element it retimes is clocked by a locally gated clock.

When this variable is set to *true*, retiming registers are clocked by the (inverted sense of) the locally gated clock signal instead.

See Also

- insert_dft

---

test_enable_scan_reordering_in_compile_incremental

Enables scan chain reordering and repartitioning when an incremental compile is performed in Design Compiler Graphical.

Data Types

Boolean

Default true

Description

This variable controls whether scan chain reordering is applied during an incremental compile in Design Compiler Graphical, which is performed by the following command:

```
prompt> compile_ultra -incremental -spg
```

The default is *true*, which enables reordering. To disable reordering, set this variable to *false*.

When reordering is enabled, repartitioning is also performed unless explicitly disabled with the *set_optimize_dft_options -repartitioning_method none* command.

Two flows are supported, as described in the following sections.

Binary DFTMAX Flow

In this flow, native DFT insertion, performed with the *insert_dft* command, is followed by an incremental topographical compile. This incremental compile performs an incremental placement of the DFT-added logic (such as reconfiguration MUXs and compression logic) When this variable is set to *true*, the incremental compile also performs scan chain reordering to minimize the wire length of the overall scan connectivity.

This flow has the following requirements and limitations:

- Binary design data must be used for the incremental compile. You must perform the incremental compile in the same session as DFT insertion, or store the design in a .ddc file between sessions.
- Only DFTMAX compressed scan designs are supported. Pure standard scan designs are not supported.
- The *set_optimize_dft_options* command is not supported.

ASCII Netlist Flow

In this flow, an ASCII netlist from a third-party DFT insertion tool is read in, along with a SCANDEF file describing the scan structures. When this variable is set to *true*, an incremental compile in this flow also performs scan chain reordering to minimize the wire length of the overall scan connectivity.

In this flow, you can use the *set_optimize_dft_options* command to configure scan reordering, just as you would in IC Compiler.

This flow has the following requirements and limitations:

- Hierarchical SCANDEF is not supported; only a flat DFT flow is supported.
- Post-DFT DRC is not supported after reordering. Perform DRC checking in the ATPG tool to verify scan chain integrity.

See Also

- compile_ultra

---

test_fast_feedthrough_analysis

Controls whether DFT DRC uses the TetraMAX fast feedthrough analysis engine.

Data Types

Boolean

Default false

Description

A *clock feedthrough* is a logic path in the design where the output pin of the path has the same clock behavior as the input pin of the path. The tool uses feedthrough path information during DRC. When the tool writes out CTL test model information for the design, the model includes any feedthrough paths that span from input to output.

Feedback

By default, DFT Compiler performs basic analysis of clock network logic to determine clock propagation behavior. However, for complex clock network logic, you can enable advanced clock feedthrough analysis by setting the following variable:

```
prompt> set_app_var test_fast_feedthrough_analysis true
```

This variable setting uses the TetraMAX DRC engine (which the *dft_drc* command runs internally) to detect clock feedthrough paths in the logic of the current design.

Enabling fast feedthrough analysis can help during both core creation and core integration:

- During core creation, feedthroughs from input port through complex logic to output port can be detected and included in the CTL model of the core.
- During core integration, if you have CTL-modeled cores that contain unidentified feedthrough paths, you can also set the *test_simulation_library* variable to configure the TetraMAX DRC engine to use a netlist simulation model for that core.

  If you created the core with advanced feedthrough analysis, the CTL model should include any feedthrough paths through complex logic and you should not need to reanalyze the logic using a netlist simulation model. However, verify that the CTL model includes all expected feedthrough paths in this case.

Note that although feedthroughs that vary on a per-test-mode basis can be understood by DFT DRC and insertion of the current design, they cannot be described in the resulting CTL model.

See Also

- dft_drc

---

test_icg_n_ref_for_dft

Defines the integrated clock-gating cell to be used for implementing return-to-one clock gating in the DFT logic.

Data Types

string

Default ""

Description

When DFT Compiler inserts clock-gating cells as a part of the DFT logic, this variable constrains the insertion process to use the specified library cell to gate any clock signal that has a return-to-one (RTO) waveform. The name must match an available cell in the target libraries, and this cell must be an integrated clock-gating cell.

Feedback

Specify the cell name without the library name. If the library name is included, the specification is ignored without warning or error.

The cell specified by this variable is used only for the initial logic construction. After insertion, subsequent design optimizations can resize the cell (if allowed by the library cell attributes).

This variable applies to DFT logic inserted by DFTMAX during on-chip clocking (OCC) controller insertion and serializer insertion.

By default, when this variable is not set, DFTMAX will gate clock signals using regular combinational gates.

See Also

- set_dft_configuration
- set_scan_compression_configuration
- set_dft_clock_controller
- test_icg_p_ref_for_dft

---

test_icg_p_ref_for_dft

Defines the integrated clock-gating cell to be used for implementing return-to-zero clock gating in the DFT logic.

Data Types

string

Default ""

Description

When DFT Compiler inserts clock-gating cells as a part of the DFT logic, this variable constrains the insertion process to use the specified library cell to gate any clock signal that has a return-to-zero (RTZ) waveform. The name must match an available cell in the target libraries, and this cell must be an integrated clock-gating cell.

Specify the cell name without the library name. If the library name is included, the specification is ignored without warning or error.

The cell specified by this variable is used only for the initial logic construction. After insertion, subsequent design optimizations can resize the cell (if allowed by the library cell attributes).

This variable applies to DFT logic inserted by DFTMAX. This includes: on-chip clocking (OCC) controllers, serializer, DFTMAX Ultra, and LogicBIST insertion.

Feedback

By default, when this variable is not set, DFTMAX will gate clock signals using regular combinational gates.

See Also

- set_dft_configuration
- set_scan_compression_configuration
- set_dft_clock_controller
- test_icg_n_ref_for_dft

---

test_infer_slave_clock_pulse_after_capture

Guides protocol inference for master/slave test design methodologies during execution of the *dft_drc* command.

Data Types

string

Default infer

Description

Guides protocol inference for master/slave test design methodologies during execution of the *dft_drc* command.

When set to *infer* (the default value), *dft_drc* protocol inference is based on an analysis of the scan cell states

The other possible values are *pulse* and *no_pulse*. When set to *pulse*, all slave clocks are pulsed after capture. When set to *no_pulse*, no slave clocks are pulsed after capture.

If a slave clock is being inferred, and the value of the variable is invalid, the warning message TEST-314 is issued and the compiler uses the *infer* value by default.

See Also

- dft_drc

---

test_input_wrapper_chain_naming_style

Specifies the naming style used for input core wrapper chains.

Data Types

string

Feedback

Default ""

Description

This variable determines the naming style to be used by the *insert_dft* command for naming input wrapper chains, which are created when the *-mix_cells* option of the *set_wrapper_configuration* command is set to *false*. (This is the default for maximized reuse core wrapping, but not for simple core wrapping.)

This variable must contain one '%s' (percent s) character sequence. The '%s' is replaced with the ordinal chain number, beginning with 1.

The '%s' must be placed at the end of the string, so that the string represents a prefix. In addition, the string '<testmode>_i_' is prefixed to the specified string.

Examples

For example, to name the input wrapper chains as IP_IWRAP_1, IP_IWRAP_2, and so on,

```
set_app_var test_wrapper_chain_naming_style {IP_IWRAP_%s}
```

See Also

- insert_dft

---

test_isolate_hier_scan_out

Prevents the *insert_dft* command from inserting logic that isolates scan connections at hierarchical boundaries during functional operation.

Data Types

integer

Default 0

Description

This variable prevents the *insert_dft* command from inserting logic that isolates scan connections at hierarchical boundaries during functional operation. When the value is set to *1*, the *insert_dft* command inserts logic that isolates scan connections at hierarchical boundaries during functional operation. This can reduce dynamic switching currents and output loading. When set to *0* (the default), no logic is inserted.

See Also

- insert_dft

Feedback

test_keep_connected_scan_en

Controls whether existing scan-enable signals are disconnected during DFT insertion.

Data Types

Boolean

Default false

Description

During DFT insertion, DFT Compiler identifies the scan-enable pins of scan cells and test models that should be connected to the global scan-enable signal. These are known as scan-enable target pins.

When this variable is set to the default value of *false*, if a scan-enable target pin already has a connection, DFT Compiler disconnects it to make the connection to the global scan-enable pin. During DFT insertion, the *insert_dft* command issues a TEST-394 warning to note the disconnection:

```
Warning: Disconnecting pin 'UMEMWRAP/UMEM/SE' to route scan enable.
 (TEST-394)
```

When this variable is set to *true*, if a target scan-enable pin already has a connection (besides a test-ready deasserted constant value), DFT Compiler skips that target pin. In this case, the *insert_dft* command issues a TEST-410 warning to confirm that the existing connection is kept:

```
Warning: Not disconnecting pin 'UMEMWRAP/UMEM/SE' to route scan enable.
 (TEST-410)
```

If you need to make a scan-enable connection somewhere else in the existing logic driving the target pin, you can use the *set_dft_connect* command to add additional scan-enable target pins for DFT insertion:

```
set_dft_signal \\
  -type ScanEnable -usage scan \\
  -view spec \\
  -port SCAN_EN -connect_to UMEMWRAP/SCAN_EN
```

During DFT insertion, DFT Compiler may replace existing scan cells. To prevent this behavior the scan-enable target pins of the scan cell must be functionally connected. Set this variable to *true* to keep this connection and avoid replacing the scan cell.

See Also

- set_dft_connect
- set_dft_signal

Feedback

test_logicbist_add_prpg_lul

Enables the insertion of lockup latches between the PRPG and the first scan-cells in the DFTMAX logicBIST scan-chains.

Data Types

Boolean

Default false

Description

When the variable *test_logicbist_add_prpg_lul* is set to *true* the insert_dft will insert lockup latches in a DFTMAX LogicBIST design between the PRPG and the first scan-cells in the scan-chains.

The lockup latches will make sure to drive the internal scan-chains in a skew safe manner in case the clock tree cannot be balanced, for example if OCCs are implemented on all clocks in the design and there is no plan to balance the asynchronous clock trees for power reason.

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

```
prompt> printvar test_logicbist_add_prpg_lul
```

See Also

- set_logicbist_configuration

test_mode_port_inverted_naming_style

Specifies the naming style to use for the *test_hold_logic_zero* type of test mode signal ports to be created in the design.

Data Types

string

Default test_mode_i%s

Description

This variable specifies the naming style to use for the *test_hold_logic_zero* type of test-mode signal ports to be created in the design. New ports are not created if suitable ports are identified with the *set_dft_signal* command.

Feedback

This variable must contain one %s (percent s) character sequence. When a new port is added, if its name is the same as that of an existing port, %s is replaced by a character (*a* through *z*) that creates a unique name.

For example, if this variable is set to *MY_TM_I%s*, when synthesis adds a new port for the test-mode signal, the port is named

MY_TM_I

If a port named *MY_TM_I* already exists in the design, the new port is named

MY_TM_Ia

See Also

- insert_dft

---

test_mode_port_naming_style

Determines the naming style to be used by the *insert_dft* command for test mode ports created in designs during the addition of test point circuitry.

Data Types

string

Default test_mode%s

Description

This variable determines the naming style to be used by the *insert_dft* command for test mode ports created in designs during the addition of test point circuitry. New ports are not created if suitable ports are identified with the *set_dft_signal* command.

This variable must contain one %s (percent s) character sequence. When a port is added, if its name is the same as that of an existing port, %s is replaced by a character (*a* through *z*) that creates a unique name.

For example, if this variable is set to *MY_TM%s*, when synthesis adds a new port for the test mode signal, the port is named

MY_TM

If a port named *MY_TM* already exists in the design, the new port is named

MY_TMa

See Also

- insert_dft

Feedback

test_mux_constant_si

Specifies how scan insertion uses a port you declare as scan input, when the port is tied high or to the ground in functional mode.

Data Types

Boolean

Default false

Description

This variable specifies how scan insertion uses a port you declare as scan input, when the port is tied high or to the ground in functional mode.

When this variable is set to *false* (the default value), scan insertion ignores the tie-off logic and directly uses the port as a scan input. This might change the output of the design during functional mode.

When this variable is set to *true*, scan insertion multiplexes the scan input signal with the constant logic, using the scan enable signal to control the multiplexer.

See Also

- insert_dft

test_mux_constant_so

Specifies how scan insertion treats specified scan output ports with existing constant connections.

Data Types

Boolean

Default false

Description

This variable specifies how scan insertion uses a port you declare as a scan output, when the port has an existing connection to logic zero or logic one before scan insertion.

When this variable is set to *false*, which is default, if the specified port is tied to logic zero or logic one, DFT Compiler ignores the constant value during scan insertion and drives the port directly as a scan output. This behavior ensures that no additional logic is added at the port if it was undriven in the RTL and tied to logic zero during synthesis.

Feedback

If the existing constant value driving the port is required for proper operation in functional mode, set this variable to *true*. In this case, DFT Compiler multiplexes the scan-out signal with the constant value, using the scan-enable signal to control the multiplexer, so that the functional behavior remains unchanged.

This variable affects only scan output ports that are explicitly defined using the *set_dft_signal -type ScanDataOut* command.

See Also

- insert_dft

---

test_non_scan_clock_port_naming_style

Specifies the style the *insert_dft* command uses to name the ports that clock gating creates for nonscan clocks.

Data Types

string

Default test_nsc_%s

Description

This variable specifies the style the *insert_dft* command uses to name the ports that clock gating creates for nonscan clocks.

This variable must contain one %s (percent s) character sequence. The %s is replaced by a string identifying the original clock, its inversion (either *i* for inverted or *n* for noninverted), and the inactive level (either 0 or 1).

The following example shows the renaming of this variable from its setting of MY_NSC_ %s to MY_NSC_clockn0. When *insert_dft* creates a nonscan clock (from an original clock named "clock") that is noninverted and inactive low, that port is named

MY_NSC_clockn0

See Also

- insert_dft

---

test_occ_insert_clock_gating_cells

Enables the use of clock-gating cells in the implementation of the on-chip clocking (OCC) controller.

Feedback

Data Types

Boolean

Default false

Description

By default, when a DFT-inserted on-chip clocking (OCC) controller is implemented, combinational multiplexer logic is used to switch between the clocks. You can set this variable to *true* to use latch-based clock-gating clock selection logic instead.

When this variable is set to *true*, the following capabilities are also available:

- If you want to use a particular integrated clock-gating (ICG) cell, you can specify it with the *test_icg_p_ref_for_dft* variable.
- If you want to use particular library cells for the clock selection logic, you can specify them with the *occ_lib_cell_nor2*, *occ_lib_cell_andor21*, *occ_lib_cell_andor22*, *occ_lib_cell_clkbuf*, and *occ_lib_cell_clkinv* design attributes (in supported combinations).
- If you want a dedicated OCC clock chain connection that exists only on the ATE clock path, you can set the *test_dedicated_clock_chain_clock* variable to *true*.

See Also

- set_dft_configuration
- set_dft_clock_controller
- test_dedicated_clock_chain_clock
- test_icg_p_ref_for_dft

---

test_occ_ip_leaf_pin

Specifies whether to allow output of a hierarchical instance to be a PLL clock.

Data Types

Boolean

Default false

Description

The argument to set_dft_clock_controller -pllclocks option are normally an output pin of a leaf cell.

Feedback

When set to true, you can specify PLL clocks to be the output of a hierarchical instance. The tool will connect these PLL clock signals to the DFT clock controller inserted by the insert_dft command.

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

```
prompt> printvar test_occ_ip_leaf_pin
```

See Also

- set_dft_signal
- set_dft_clock_controller

---

test_optimize_dft_ng

Enables the next generation engines for scan chain reordering and repartitioning in Design Compiler.

Data Types

Boolean

Default false

Description

This variable controls whether the new or the previous engines are used for scan chain reordering and repartitioning in Design Compiler.

The default is *false*. To enable the next generation engines, set this variable to *true*.

See Also

- compile_ultra
- set_optimize_dft_options
- test_enable_scan_reordering_in_compile_incremental

---

test_output_wrapper_chain_naming_style

Specifies the naming style used for output core wrapper chains.

Data Types

string

Default ""

Feedback

Description

This variable determines the naming style to be used by the *insert_dft* command for naming output wrapper chains, which are created when the *-mix_cells* option of the *set_wrapper_configuration* command is set to *false*. (This is the default for maximized reuse core wrapping, but not for simple core wrapping.)

This variable must contain one '%s' (percent s) character sequence. The '%s' is replaced with the ordinal chain number, beginning with 1.

The '%s' must be placed at the end of the string, so that the string represents a prefix. In addition, the string '<testmode>_o_' is prefixed to the specified string.

Examples

For example, to name the output wrapper chains as IP_OWRAP_1, IP_OWRAP_2, and so on,

```
set_app_var test_wrapper_chain_naming_style {IP_OWRAP_2}
```

See Also

- insert_dft

---

test_preview_scan_shows_cell_types

Shows cell instance types when running the *preview_dft* command.

Data Types

string

Default false

Description

This variable, when set to *true*, shows cell instance types when running the *preview_dft* command. By default, cell type information is suppressed.

See Also

- preview_dft

---

test_rtldrc_latch_check_style

Specifies the latch check style to use during *rtldrc* command activities.

Feedback

Data Types

string

Default default

Description

Specifies the latch check style to use during *rtldrc* command activities. The value of this variable affects the outcome of the *rtldrc* command, which uses this variable to determine the way latches are checked for violations against the design rules of a scan test methodology. When latch transparency checks are selected, nontransparent latches are flagged as violations (*TEST-1211*). This also suppresses checking for hold data violations (*TEST-965*) and latch controllability violations (*TEST-1210*).

To determine the current value of this variable, enter one of the following commands, depending on which mode you are using:

```
prompt> printvar test_rtldrc_latch_check_style
or
dc_shell-t> printvar test_rtldrc_latch_check_style
```

See Also

- set_scan_configuration

---

test_scan_chain_naming_style

Specifies the naming style used for non-wrapper scan chains.

Data Types

string

Default ""

Description

This variable determines the naming style to be used by the *insert_dft* command for naming scan chains. (In core-wrapped designs, wrapper chains are named using separate variables.)

This variable must contain one '%s' (percent s) character sequence. The '%s' is replaced with the ordinal chain number, beginning with 1.

The '%s' must be placed at the end of the string, so that the string represents a prefix.

The default is the ordinal chain number, beginning with 1.

Feedback

Examples

For example, to name the scan chains as IP_SCAN_1, IP_SCAN_2, and so on,

```
set_app_var test_scan_chain_naming_style {IP_SCAN_%s}
```

See Also

- insert_dft

---

test_scan_clock_a_port_naming_style

Determines the naming style to be used by the *insert_dft* command for test scan clock a ports created in designs during the addition of test scan circuitry.

Data Types

string

Default test_sca%s

Description

This variable determines the naming style to be used by the *insert_dft* command for test scan clock a ports created in designs during the addition of test scan circuitry. This variable is used in the same way as the *test_clock_port_naming_style* variable is used. If the *set_dft_signal* command identifies suitable ports, new ports are not added.

This variable must contain one %s (percent s) character sequence. When adding a port, if the new port name is the same as an existing port name, %s is replaced by a character (*a* through *z*) that creates a unique name.

For example, if this variable is set to *MY_TSCA%s* when *insert_dft* adds a new port for the *test scan clock a* signal, the port is named

MY_TSCA

If a port named *MY_TSCA* already exists in the design, the new port is named

MY_TSCAa

See Also

- insert_dft
- test_clock_port_naming_style
- test_scan_clock_port_naming_style
- test_scan_clock_b_port_naming_style

Feedback

- test_scan_enable_inverted_port_naming_style
- test_scan_enable_port_naming_style

---

test_scan_clock_b_port_naming_style

Determines the naming style to be used by the *insert_dft* command for test scan clock b ports created in designs during the addition of test scan circuitry.

Data Types

string

Default test_scb%s

Description

This variable determines the naming style to be used by the *insert_dft* command for test scan clock b ports created in designs during the addition of test scan circuitry. This variable must contain one %s (percent s) character sequence. When adding a port, if the new port name is the same as an existing port name, %s is replaced by a character (*a* through *z*) that creates a unique name.

For example, if this variable is set to *MY_TSCB%s*, when *insert_dft* adds a new port for the *test scan clock b* signal, the port is named

MY_TSCB

If a port named *MY_TSCB* already exists in the design, the new port is named

MY_TSCBa

See Also

- insert_dft
- set_dft_signal
- test_clock_port_naming_style
- test_scan_clock_port_naming_style
- test_scan_clock_a_port_naming_style
- test_scan_enable_inverted_port_naming_style
- test_scan_enable_port_naming_style

Feedback

test_scan_clock_port_naming_style

Determines the naming style used by the *insert_dft* command for global test scan signal ports created in designs during the addition of test circuitry.

Data Types

string

Default test_sc%s

Description

This variable determines the naming style used by the *insert_dft* command for global test scan signal ports created in designs during the addition of test circuitry. This variable is used in the same way as the *test_clock_port_naming_style* variable is used. This variable must contain one %s (percent s) character sequence. When adding a port, if the new port name is the same as an existing port name, %s is replaced by a character (*a* through *z*) that creates a unique name.

For example, if this variable is set to *MY_TSC%s* when *insert_dft* adds a new port for the *test scan clock* signal, the port is named

MY_TSC

If a port named *MY_TSC* already exists in the design, the new port is named

MY_TSCa

See Also

- insert_dft
- set_dft_signal
- test_clock_port_naming_style
- test_scan_clock_a_port_naming_style
- test_scan_clock_b_port_naming_style
- test_scan_enable_inverted_port_naming_style
- test_scan_enable_port_naming_style

test_scan_enable_inverted_port_naming_style

Determines the naming style to be used by the *insert_dft* command for scan-enable inverted ports created in designs during the addition of test scan circuitry.

Feedback

Data Types

string

Default test_sei%s

Description

This variable determines the naming style to be used by the *insert_dft* command for scan-enable inverted ports created in designs during the addition of test scan circuitry. This variable is used in the same way as the *test_clock_port_naming_style* variable is used. This variable must contain one %s (percent s) character sequence. When adding a port, if the new port name is the same as an existing port name, %s is replaced by a character (*a* through *z*) that creates a unique name.

For example, if this variable is set to *MY_TSEI%s* when *insert_dft* adds a new port for the *test scan enable inverted* signal, the port is named

MY_TSEI

If a port named *MY_TSEI* already exists in the design, the new port is named

MY_TSEIa

See Also

- insert_dft
- set_dft_signal
- test_clock_port_naming_style
- test_scan_clock_port_naming_style
- test_scan_clock_a_port_naming_style
- test_scan_clock_b_port_naming_style
- test_scan_enable_port_naming_style

---

test_scan_enable_port_naming_style

Determines the naming style to be used by the *insert_dft* command for test scan-enable ports created in designs during the addition of test scan circuitry.

Data Types

string

Default test_se%s

Feedback

Description

This variable determines the naming style to be used by the *insert_dft* command for scan-enable ports created in designs during the addition of test scan circuitry. This variable is used in the same way as the *test_clock_port_naming_style* variable is used. This variable must contain one %s (percent s) character sequence. When adding a port, if the new port name is the same as an existing port name, %s is replaced by a character (*a* through *z*) that creates a unique name.

For example, if this variable is set to *MY_TSE%s* when *insert_dft* adds a new port for the *test scan enable* signal, the port is named

MY_TSE

If a port named *MY_TSE* already exists in the design, the new port is named

MY_TSEa

In the pipelined scan-enable flow, this variable affects the scan-enable signal (of type *ScanEnable*) but not the pipeline-enable signal (of type *LOSPipelineEnable*).

See Also

- insert_dft
- set_dft_signal
- test_clock_port_naming_style
- test_scan_clock_port_naming_style
- test_scan_clock_a_port_naming_style
- test_scan_clock_b_port_naming_style
- test_scan_enable_inverted_port_naming_style

---

test_scan_in_port_naming_style

Specifies the naming style used by the *insert_dft* command for serial test-signal ports created in designs during the addition of test circuitry.

Data Types

string

Default test_si%s%s

Feedback

Description

This variable specifies the naming style used by the *insert_dft* command for serial test-signal ports created in designs during the addition of test circuitry. New ports are not added if suitable ports are identified with the *set_dft_signal* command.

This variable must contain two %s (percent s) character sequences. For a design with multiple scan chains, the index of the connecting scan chain replaces the first %s. For designs with single scan chains, %s is replaced by an empty string (""). The second %s is replaced by an alphabetic character if the new port name conflicts with an existing name.

For example, if this variable is set to *SI_%s%s*, when *insert_dft* adds a new port for the only scan-in signal of a design with one scan chain, the port is named

SI

If this variable is set to *TDI_%s%s*, when *insert_dft* adds a new port for the scan-in signal of the second scan chain in the design, the port is named

TDI_2

See Also

- insert_dft
- test_scan_out_port_naming_style

---

test_scan_link_so_lockup_key

Indicates to the *preview_dft* command what key to use to identify cells with scan-out lock-up latches in reports.

Data Types

string

Default l

Description

This variable specifies to the *preview_dft* command what key to use to identify cells with scan-out lock-up latches in reports.

See Also

- preview_dft

Feedback

test_scan_link_wire_key

Indicates to the *preview_dft* command the key to use to identify cells that drive wire-scan links in reports.

Data Types

string

Default w

Description

This variable indicates to the *preview_dft* command the key to use to identify cells that drive wire-scan links in reports.

See Also

- preview_dft

test_scan_out_port_naming_style

Specifies the naming style used by the *insert_dft* command serial test-signal ports created in designs during the addition of test circuitry.

Data Types

string

Default test_so%s%s

Description

This variable specifies the naming style used by the *insert_dft* command for serial test-signal ports created in designs during the addition of test circuitry. New ports are not added if suitable ports are identified with the *set_dft_signal* command.

This variable must contain two %s (percent s) character sequences. For a design with multiple scan chains, the index of the connecting scan chain replaces the first %s. For designs with single scan chains, %s is replaced by an empty string (""). The second %s is replaced by an alphabetic character if the new port name conflicts with an existing name.

For example, if this variable is set to *SO_%s%s*, when *insert_dft* adds a new port for the only scan-out signal of a design with one scan chain, the port is named

SO

If this variable is set to *TDO_%s%s*, when *insert_dft* adds a new port for the scan-out signal of the second scan chain in the design, the port is named

Feedback

TDO_2

See Also

- insert_dft
- test_scan_in_port_naming_style

---

test_scan_segment_key

Instructs the *preview_dft* command what key to use to identify scan segments in reports.

Data Types

string

Default s

Description

This variable instructs the *preview_dft* command what key to use to identify scan segments in reports.

See Also

- preview_dft

---

test_scan_segment_physical_location_checks_skip

Allows to only report missing physical placement coordinates of segment componets avoiding the early and sudden preview_dft and insert_dft ending.

Data Types

Boolean

Default false

Description

During DFT insertion, when dealing with component physical placement DFT compiler needs the XY component physical coordinates. If these XY coordinates are not supplied, there will be shown a "TEST-1016" message and preview_dft and insert_dft will fail in rare situations. This variable offers to bypass this warning and behavior setting it to true under the user responsability. If set to false, normal behavior and warning will happen.

Setting of this variable must be done at the beginning of the main TCL script.

Feedback

See Also

- TEST-1016
- insert_dft
- preview_dft

---

test_scan_true_key

Specifies to the *preview_dft* command the key to use to identify cells with true-scan attributes in reports.

Data Types

string

Default t

Description

This variable specifies to the *preview_dft* command the key to use to identify cells with true scan attributes in reports.

See Also

- preview_dft

---

test_scandef_stop_skip_last_segment_with_scanout_lockup

Keeps the SCANDEF chains from having a lockup latch as STOP point.

Data Types

Boolean

Default false

Group

test_variables

Description

When activated, the SCANDEF information generated by the *insert_dft* command will not have a lockup latch as a STOP point.

Feedback

See Also

- insert_dft
- write_scan_def
- check_scan_def

---

test_serialize_put_fsm_clock_output

Allows the serializer clock controller outputs to be described in the STIL protocol file.

Data Types

Boolean

Default true

Description

By default, when DFTMAX inserts a serializer, it includes information about the outputs of the serializer clock controller in the STIL protocol file for ScanCompression_mode. The information is placed in a ClockStructure block. This information helps TetraMAX trace the source of the signal clock signals during DRC.

You can prevent the addition of this information by setting this variable to *false*.

See Also

- set_dft_configuration
- set_scan_compression_configuration

---

test_set_svf_print_exclude_existing_dft_se_alike

test_set_svf_print_exclude_existing_dft_se_alike

Data Types

bool

Default false

Description

The default behavior of set_svf -print path is to automatically include in the guide_scan_input of the svf file all signals defined as either set_dft_signal -view spec or set_dft_signal -view existing_dft. This behavior for the second case is not correct since already-existing test signal guidance should be specified using set_dft_signal -active_state [0|1].

The default behavior might lead to problems latter in the verification step. The variable *test_set_svf_print_exclude_existing_dft_se_alike* fixes this issue by including only the signals defined using set_dft_signal -view spec.

See Also

- set_dft_signal
- set_svf

---

test_setup_additional_clock_pulse

Adds an extra clock cycle to all clocks in the design in the initialization procedure during the execution of the *create_test_protocol* command.

Data Types

Boolean

Default false

Description

When this variable is set to *true*, an extra clock cycle is added to all clocks in the design in the initialization procedure during the execution of the *create_test_protocol* command.

Set this variable to *true* if you are using clock gating and if the off-state of the clock driving the clock gating (at time=0) is the opposite of the active state of the clock-gating latch enable input.

The extra cycle ensures that all clock-gating latches are at a known state at the end of the initialization procedure.

See Also

- create_test_protocol
- dft_drc
- read_test_protocol
- write_test_protocol

---

test_shared_codec_io_architecture

Specifies whether a consistent load-mode architecture should be used for DFTMAX codec insertion in the shared codec I/O flow.

Feedback

Data Types

Boolean

Default false

Description

This variable specifies whether codecs inserted by DFTMAX use a consistent load-mode architecture, which can be helpful in the shared codec I/O flow.

When the shared codec I/O feature is enabled with the *-shared_inputs* option of the *set_scan_compression_configuration* command, the tool shares three categories of input pins across all codecs: load-mode, high X-tolerance, and scan pins. Pins are shared across codecs within each category, but not across the categories.

The high X-tolerance and scan pin counts are predictable, but the load-mode pin count can sometimes vary across codecs, which can make it difficult to know what sharing configuration to use.

You can set this variable to *true* to force newly inserted codecs in the current design to use two load-mode pins. This prevents load-mode pin count variability.

If you leave this variable set to its default of *false*, load-mode pins are still shared, even if their counts differ across codecs.

For more information on shared codec I/O pin categories, see the user guide documentation on determining the fully shared I/O configuration.

See Also

- set_scan_compression_configuration

---

test_simulation_library

Specifies the path name to a Verilog simulation library.

Data Types

list

Description

The *test_simulation_library* variable specifies the file names of one or more Verilog simulation libraries used by the *dft_drc* command. These simulation models are necessary if you have cells which have no functional information in the technology library, such as memories.

If a simulation model is provided for a cell which already has functional information in the technology library, the simulation model takes precedence during *dft_drc*. This allows

Feedback

complete simulation models to be provided for complex cells which have simplified or dummy functionality in the technology library.

The Verilog functional descriptions for the cells specified with the *test_simulation_library* variable must be self-contained. They cannot depend on any functionality of cells defined in the $link_library, since *dft_drc* does not convert the entire $link_library; it only converts the cells instantiated by the current design.

Examples

The following example shows how you can set the simulation library to files containing multiple models each:

```
lappend search_path ./my_lib_models
set test_simulation_library {all_my_memories.v all_my_reg_files.v}
```

In the above example, the memory and register file model files are located using the search path, and read in to comprise the simulation library.

The *test_simulation_library* variable also accepts the asterisk (*) wildcard character in the file names to allow for the case where multiple files under a certain directory comprise the simulation library files:

```
set test_simulation_library {/root/install/xyz/lib/verilog/*.v}
```

In this example, all files with a ".v" name extension under the /root/install/xyz/lib/verilog/ directory are considered to comprise the simulation library. The asterisk (*) is the only wildcard character allowed in the library file name. If a wildcard is used in the file name, specify the full path name of the library files. This avoids the combined effect of the search_path and the wildcard character bringing in unexpected files as the simulation library.

See Also

- dft_drc

---

test_soc_core_wrap_allow_multibit_ioregs

Specifies whether multibit registers can be shared wrapper cells in maximized reuse wrapper chains.

Data Types

Boolean

Default true

Feedback

Description

This variable specifies whether multibit registers can be shared wrapper cells in wrapper chains.

When set to *true* (the default), multibit cells can be used in wrapper chains. For best results, configure core wrapping before performing multibit banking with

- The initial *compile_ultra* command (RTL inference flow)
- The *identify_register_banks* command (physical banking flow)

This ensures that multibit cells anticipate the I/O register classification requirements of core wrapping.

When set to *false*, multibit cells cannot be used in wrapper chains. Note that in this case, core wrapping does not place any restrictions on multibit banking of I/O registers. If any port is associated with a multibit cell, a TEST-1067 warning message is issued and the port gets a dedicated wrapper cell. If you see this message, you can improve your QoR by restoring this variable to its default of *true*.

This variable is used only when the maximized reuse wrapper flow is enabled. Multibit cells cannot be shared wrapper cells in the simple wrapper flow.

---

test_soc_wrp_soft_core_non_abutted_flow

Allows user to enable non abutted wrapping.

Data Types

Boolean

Default false

Description

This variable allows user to enable non-abutted flow in partition aware soft core testcase. Non-abutted flow allows wrapper analysis of top level along with soft cores. In order to see this feature define partitions using *define_partition* command. Enable soft cores using *set_wrapper_configuration -wrapper enable*. And enable this variable by *set test_soc_wrp_soft_core_non_abutted_flow true* command.

Examples

For example, define_partition P1 -wrapper enable set_wrapper_configuration preview_dft insert_dft

```
set test_soc_wrp_soft_core_non_abutted_flow true
```

Feedback

See Also

- insert_dft

---

test_stil_max_line_length

Specifies the maximum line length for the file written by the *write_test_protocol -format stil* command.

Data Types

integer

Default 72

Description

This variable specifies the maximum line length for the file written by the *write_test_protocol -format stil* command. The recognized value is any integer between 60 and 5000. The default value is 72.

When *write_test_protocol -format stil* writes the *design_name*.spf file, the command inserts a new line at or before the value of this variable, but identifiers and keywords are always preserved. If an identifier is longer than the value of this variable, the command makes an exception to the line length limit so as to preserve the identifier. The identifier begins at a new line and is printed in its entirety before the next new line.

See Also

- write_test_protocol

---

test_suppress_toggling_instance_name_prefix

Specifies the prefix used for toggle suppression gates inserted at scan cell functional outputs.

Data Types

string

Default ""

Description

This variable specifies the prefix used in generating cell instance names for toggle suppression gates inserted by the *insert_dft* command at scan cell functional outputs.

Feedback

The variable has an effect only when used in conjunction with the *set_scan_suppress_toggling* command, which configures the insertion of toggle suppression logic that prevents unnecessary functional logic toggling during scan shift.

When toggling suppression is enabled, and the variable is set to a string value "*myprefix*", toggle suppression gates added by the *insert_dft* command will be named in the form:

```
<myprefix>_<register_cell_leaf_name>_<gated_output_pin_name>
```

For example, if you set the variable to QGATE, a gating cell inserted at CORE/ENAB_reg/Q would be named CORE/QGATE_ENAB_reg_Q. Since the gating cell is placed at the same level of hierarchy as the gated scan cell, they both share the same hierarchical path name.

When this variable is set to an empty string, which is the default, the gating cell instance names are determined by the *compile_instance_name_prefix* variable.

To determine the current value of this variable, use the *printvar test_suppress_toggling_instance_name_prefix* command. For a list of compile variables and their current values, use the *print_variable_group test* command.

See Also

- set_scan_suppress_toggling

---

test_sync_occ_1x_period

Specifies the clock period for the slowest-frequency PLL clock of a DFT-inserted synchronous OCC controller.

Data Types

positive nonzero integer

Default 20

Description

When you insert a synchronous OCC controller, you can optionally specify the clock period of the slowest-frequency clock with this variable. It affects the clock period values in the ClockTiming block of the STIL protocol file created by the tool. Although the value does not affect pattern generation in TetraMAX, you can specify it for informational purposes.

This value is used for all synchronous OCC controllers in the design.

This variable must be set so that any 2X or 4X clocks have an integer period value greater than 1 when derived (by division) from this 1X clock period value. If you control any 2X frequency clocks, this value should be at least 4 and be divisible by 2. If you control any 4X frequency clocks, this value should be at least 8 and be divisible by 4.

Feedback

This variable pertains only to synchronous OCC controllers, whose clocks are specified with the *-1x_clocks*, *-2x_clocks*, and *-4x_clocks* options of the *set_dft_clock_controller* command.

See Also

- set_dft_clock_controller

---

test_tp_enable_logic_type

Specifies what type of enable logic to implement for test point registers.

Data Types

Boolean

Default gate

Description

This variable controls what type of enable logic to implement for test point registers.

When a test point's control signal is deasserted, its test point registers is held at a known static value. This variable controls how that logic is implemented.

The valid values for this variable are:

- *gate* - AND/OR gate at the register Q pin

  The register output is ANDed or ORed with the control signal. When the control signal is deasserted, the register output is also deasserted.

- *reset* - asynchronous-reset register

  The register is reset to logic zero whenever the control signal is deasserted. (However, the reset is blocked when the scan-enable signal is asserted, to allow for scan operation in test modes where the test point is not active.)

- *set* - asynchronous-set register

  This is the same as the *reset* case, except that an asynchronous set is implemented instead.

The logic implementation considers whether the control signal is active-high or active-low, and the type of enable logic specified by this variable, so that deasserting the control signal deactivates the test points.

This variable applies to both automatically inserted and user-defined test point registers. It requires that the *testability* DFT client be enabled with the *set_dft_configuration* command.

Feedback

See Also

- preview_dft
- set_testability_configuration
- set_test_point_element

---

test_user_defined_instruction_naming_style

Indicates to the *check_bsd* command and the *write_bsdl* command the naming style to use for the user-defined (nonstandard) instructions inferred by these commands.

Data Types

string

Default false

Description

This variable indicates to the *check_bsd* command and the *write_bsdl* command the naming style to use for the user-defined (nonstandard) instructions inferred by these commands. The format for the specification is *string*%d; the specification must contain exactly one %d. Names resulting from the default specification (*USER%d*) are USER1, USER2, and so on.

See Also

- check_bsd
- write_bsdl

---

test_user_test_data_register_naming_style

Indicates to the *check_bsd* command and the *write_bsdl* command the naming style to use for the user-defined (nonstandard) test data registers inferred by these commands.

Data Types

string

Default UTDR%d

Description

This variable indicates to the *check_bsd* command and the *write_bsdl* command the naming style to use for the user-defined (nonstandard) test data registers inferred by these commands. The format for the specification is *string*%d; the specification must contain

exactly one %d. Names resulting from the default specification (*UTDR%d*) are UTDR1, UTDR2, and so on.

See Also

- check_bsd
- write_bsdl

---

test_validate_test_model_connectivity

Instructs the *dft_drc* command to check the connectivity of test mode and constant pins of design instances that are represented by CTL models.

Data Types

Boolean

Default false

Description

This variable, when set to *true*, instructs the *dft_drc* command to check the connectivity of test mode and constant pins of design instances that are represented by CTL models.

When this feature is enabled, the *dft_drc* command simulates the test_setup procedure of the all_dft mode of the CTL mode associated with the current design, and then inspects the logical value at the test mode and constant pins of all instantiated designs that are represented by CTL models. The feature then reports any mismatch on a pin in term of actual value (propagated to the pin) versus the expected value (the one that is defined in the test_setup of the instantiated CTL model).

If any mismatch is found, all scan chains in the involved CTL model instance will not be considered in building scan chains in the current design.

See Also

- dft_drc
- use_test_model

---

test_wrapper_chain_naming_style

Specifies the naming style used for mixed (input/output) core wrapper chains.

Data Types

string

Feedback

Default ""

Description

This variable determines the naming style to be used by the *insert_dft* command for naming mixed (input/output) wrapper chains, which are created when the *-mix_cells* option of the *set_wrapper_configuration* command is set to *true*. (This is the default for simple core wrapping, but not for maximized reuse core wrapping.)

This variable must contain one '%s' (percent s) character sequence. The '%s' is replaced with the ordinal chain number, beginning with 1.

The '%s' must be placed at the end of the string, so that the string represents a prefix. In addition, the string '<testmode>_' is prefixed to the specified string.

Examples

For example, to name the wrapper chains as IP_WRAP_1, IP_WRAP_2, and so on,

```
set_app_var test_wrapper_chain_naming_style {IP_WRAP_%s}
```

See Also

- insert_dft

---

test_wrapper_new_wrp_clock_timing

Defines the test clock waveform for DFT-created dedicated wrapper clock signals.

Data Types

list of positive nonzero integers

Default ""

Description

This variable specifies the rising-edge and falling-edge times of any dedicated wrapper clock signals created by DFT insertion.

The values are provided in list form, just as with the *-timing* option of the *set_dft_signal* command. For example,

```
prompt> set_app_var test_wrapper_new_wrp_clock_timing {50 70}
```

Note the following requirements:

- The rising and falling edge times must be less than the clock period.
- Both edges must be after the strobe time.

Feedback

The default is an empty string (""), which results in a 10 percent duty cycle (with the rising edge and falling edge at 45 percent and 55 percent of the default clock period, respectively).

To explicitly define your own wrapper clock signal instead of using a DFT-created signal, use the *set_dft_signal* command to define a signal with the *wrp_clock* type, as shown in the following example:

```
prompt> set_dft_signal -view existing_dft -type wrp_clock -port
 MY_wrp_clk -timing {50 70}

prompt> set_dft_signal -view spec -type wrp_clock -port MY_wrp_clk
```

See Also

- test_default_period
- test_default_strobe
- test_default_strobe_width

---

test_wrp_new_power_domain_aware_dw_insertion_flow

Specifies whether new upf aware wrapper cell insertion is enabled or disabled in dft insertion flow.

Data Types

Boolean

Default false

Description

This variable is used to turn on or off new upf aware wrapper cell insertion in dft insertion flow.

When set to *false* (the default), wrapper cell insertion is based on variaous other core wrapper variables.

When set to *true*, new upf aware wrapper cell insertion is enabled in dft insertion. In this flow, tool inserts dedicated wrapper cells based on guidance from UPF APIs. Tool will not insert a dedicated wrapper cell if the cell insertion introduces non fixable MV violations.

The variable doesn't have any effect when set_wrapper_configuration option -add_wrapper_cells_to_power_domains is not enabled.

Feedback

Examples

To enable new upf aware wrapper cell insertion,

```
prompt> set_wrapper_configuration -class core_wrapper
 -add_wrapper_cells_to_power_domains enable -test_mode all_dft
prompt> set_app_var test_wrp_new_power_domain_aware_dw_insertion_flow
 true
```

---

text_editor_command

Specifies the command that executes when the *Edit/File* menu is selected in the Design Analyzer text window.

Data Types

string

Default xterm -fn 8x13 -e vi %s & in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Specifies the command that executes when the *Edit/File* menu is selected in the Design Analyzer text window.

To determine the current value of this variable, type *printvar text_editor_command*. For a list of all *view* variables and their current values, use the *print_variable_group view* command.

---

text_print_command

Specifies the command that executes when the *File/Print* menu is selected in the Design Analyzer text window.

Data Types

string

Default lpr -Plw

Description

Specifies the command that executes when the *File/Print* menu is selected in the Design Analyzer text window.

To determine the current value of this variable, use *printvar text_print_command*. For a list of all *view* variables and their current values, use the *print_variable_group view* command.

---

tieoff_hierarchy_opt

Controls whether to allow tie-off optimization down to the lowest hierarchy without port punching during fixing DRC violations of constant nets that cross multiple hierarchies.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

false in DC Explorer

Description

In RMgen reference methodology, tie-off optimization is called at the end of the *place_opt* or *psynopt* command to fix the DRC violations of constant nets that have maximum capacitance or maximum fanout.

Setting this variable to true allows tie-off optimization down to the lowest hierarchy without port punching to fix constant nets that cross multiple hierarchies.

For example, a violating constant net that drives an input port followed by multiple loads, the tool fixes the violation by inserting an optimized number of tie cells down to the lowest hierarchy to drive the loads and removing the original driving cell. You can choose to keep the original driving cell by setting the *tieoff_hierarchy_opt_keep_driver* variable.

This variable does not support multivoltage or multicorner-multimode designs. It also skips the constant nets that have the dont_touch attribute.

For multicorner-multimode designs, this variable uses information from the current scenario only.

See Also

- tieoff_hierarchy_opt_keep_driver
- set_fix_multiple_port_nets

---

tieoff_hierarchy_opt_keep_driver

Controls whether to keep the original driving cell when propagating tie-off optimization down to the lowest hierarchy.

Feedback

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

false in DC Explorer

Description

When the *tieoff_hierarchy_opt* variable is set to true, tie-off optimization is allowed to propagate to the lowest hierarchy to fix constant nets that cross multiple hierarchies. By default, the original driving cell of the constant net is removed.

This variable is used to control whether to remove the original driving cell of the constant net. When you set it to true, the original driving cell is not removed.

This variable does not support multivoltage or multicorner-multimode designs. For multicorner-multimode designs, this variable uses the information from the current scenario only.

See Also

- tieoff_hierarchy_opt

---

timing_check_defaults

Defines the default check list in the *check_timing* command.

Data Types

string

Default generated_clock loops no_input_delay unconstrained_endpoints pulse_clock_cell_type no_driving_cell partial_input_delay

Description

This variable defines the default checks to be performed when the *check_timing* command is run without any options. The default check list defined by this variable can be overridden by redefining the check list. The check list modified using the *-override_defaults*, *-include*, or *-exclude* option of *check_timing* is valid in only one command.

Note that this variable will not check if the value is correct or not; the check is done by the *check_timing* command. Each element in the check list can be one of the following strings:

```
loops
no_input_delay
```

Feedback

```
unconstrained_endpoints
generated_clock
pulse_clock_cell_type
clock_crossing
data_check_multiple_clock,
data_check_no_clock
multiple_clock
generic
gated_clock
ideal_timing
retain
clock_no_period
```

Examples

The following example defines the value of the *timing_check_defaults* variable:

```
prompt> set timing_check_defaults {clock_crossing loops}
```

See Also

- check_timing

---

timing_clock_gating_propagate_enable

Allows the gating-enable signal delay to propagate through the gating cell.

Data Types

Boolean

Default false

Description

This variable, when set to *true*, 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.

See Also

- set_clock_gating_style

Feedback

timing_clock_reconvergence_pessimism

Selects signal transition sense matching for computing clock reconvergence pessimism removal.

Data Types

string

Default normal

Description

This variable determines how the value of the clock reconvergence pessimism removal (CRPR) is computed with respect to transition sense. Allowed values are *normal* (the default) and *same_transition*.

When set to *normal*, the CRPR value is computed even if the clock transitions to the source and destination latches are in different directions on the common clock path. It is computed separately for rise and fall transitions and the value with the smaller absolute value is used.

When set to *same_transition*, the CRPR value is computed only when the clock transition to the source and destination latches have a common path and the transition is in the same direction on each pin of the common path. Thus, if the source and destination latches are triggered by different edge types, CRPR is computed at the last common pin at which the launch and capture edges match.

If the variable is set to *same_transition*, the CRPR for all minimum pulse width checks will be zero, as they are calculated using different (rise and fall) clock edges.

See Also

- report_timing
- timing_crpr_remove_clock_to_data_crp
- timing_crpr_threshold_ps
- timing_remove_clock_reconvergence_pessimism

timing_consider_internal_startpoints

Determines whether to report timing paths that start from internal startpoints.

Data Types

Boolean

Feedback

Default true

Description

When this variable is set to *true* (the default), timing paths that start from internal startpoints are reported by the *report_timing* command. When this variable is set to *false*, timing analysis skips the internal startpoints, so these paths are not reported.

Internal startpoints can result from unresolved references, cells with disabled timing arcs, or input ports with only the rising-edge or only the falling-edge input delay specified with the *set_input_delay* command.

See Also

- report_timing
- get_timing_paths
- set_input_delay

---

timing_crpr_remove_clock_to_data_crp

Allows the removal of clock reconvergence pessimism (CRP) from paths that fan out directly from the clock source to the data pins of sequential devices.

Data Types

Boolean

Default false

Description

When this variable is set to *true*, clock reconvergence pessimism (CRP) will be removed for all paths that fan out directly from the clock source pins to the data pins of sequential devices. All sequential devices that reside in the fanout of clock source pins must be handled separately in the subsequent timing update. This might cause a severe performance degradation to the timing update.

See Also

- timing_remove_clock_reconvergence_pessimism

---

timing_crpr_remove_muxed_clock_crp

Allows clock reconvergence pessimism removal (CRPR) to consider common path reconvergence between related clocks.

Feedback

Data Types

Boolean

Default true

Description

This variable controls the clock reconvergence pessimism removal (CRPR) in cases where two related clocks reconverge in the logic. Two clocks are related if one is a generated clock and the other is its parent, or both are generated clocks of the same parent clock. Although this variable name refers specifically to multiplexers, the variable applies to any situation where two related clocks reconverge within combinational logic.

By default, the separate clock paths up to the multiplexer are treated as reconvergent, and the CRP includes the reconvergence point as well as any downstream common logic. When this variable is set to *false*, the common pin is the last point where the clocks diverged to become related clocks.

If the design contains related clocks that switch dynamically, set this variable to *false* so the CRP is not removed. Related clocks switch dynamically when a timing path launches from one related clock and the clock steering logic switches dynamically so the path captures on the other related clock.

The default value is *true*, which removes the additional CRP.

See Also

- timing_remove_clock_reconvergence_pessimism

---

timing_crpr_threshold_ps

Specifies the amount of pessimism that clock reconvergence pessimism removal (CRPR) is allowed to leave in the report.

Data Types

float

Default 5

Description

This variable specifies the amount of pessimism that clock reconvergence pessimism removal (CRPR) is allowed to leave in the report. The unit is in pico seconds (ps), regardless of the units of the main library.

The threshold is per reported slack: setting this variable to the value of *TH1* means that reported slack is no worse than *S* - *TH1*, where *S* is the reported slack when the

Feedback

timing_crpr_threshold_ps variable is set close to zero (the minimum allowed value is *2e-5* pico seconds).

The variable has no effect if CRPR is not active; the *timing_remove_clock_reconvergence_pessimism* variable is set to *false*). The larger the value of *timing_crpr_threshold_ps*, the faster the runtime when CRPR is active. The recommended setting is about half stage (gate plus net) delay of a typical gate in the clock network. It provides a reasonable trade-off between accuracy and runtime in most cases.

You might want to use different settings throughout the design cycle: larger during the design phase, smaller for sign-off. You might want to experiment and set a different value when moving to a different technology.

See Also

- timing_remove_clock_reconvergence_pessimism

---

timing_disable_cond_default_arcs

Disables the default, nonconditional timing arc between pins that do have conditional arcs.

Data Types

Boolean

Default false

Description

This variable disables nonconditional timing arcs between any pair of pins that have at least one conditional arc. By 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.

See Also

- report_disable_timing

Feedback

timing_disable_recovery_removal_checks

Controls whether the tool accepts recovery and removal arcs specified in the technology library.

Data Types

Boolean

Default true

Description

This variable, when set to *false*, enables the acceptance of recovery and removal arcs specified in the technology library. Recovery or removal timing arcs impose constraints on asynchronous pins of sequential cells. Typically, recovery time specifies the time the inactive edge of the asynchronous signal has to arrive before the closing edge of the clock. Removal time specifies the length of time the active phase of the asynchronous signal must be held after the closing edge of clock.

To enable the *compile*, *report_timing*, and *report_constraint* commands to accept recovery or removal arcs specified in the library, set *timing_disable_recovery_removal_checks* to *false*.

Note that independent of the value of this variable, the *write_timing* and *report_delay_calculation* commands always accept and report recovery or removal timing information.

This variable is the logical opposite of the variable *enable_recovery_removal_arcs*. If you set either one of these variables to *false*, the tool automatically sets the other variable to *true*, and vice versa.

See Also

- compile
- enable_recovery_removal_arcs

timing_dont_traverse_pg_net

Normally, timing analysis will be processed only upon signal network and ignore Power/Ground pins. However, command *derive_pg_connection* might bring some netlist changes and make PG nets being connected to normal signal network.

In that case, timing analysis will pay huge runtime for traverse in PG networks and then bring some runtime issues. Also, case-analysis needs to honor the logic-constants bounding to PG Power/Ground.

Feedback

Data Types

Boolean

Default true

Description

This variable is used to configures whether timing analysis should touch PG networks or not. By default, timing analysis will touch PG networks and this could bring some runtime issues.

When this variable is set to *true*, timing analysis will skip traverse in PG networks and then save runtime. Meanwhile, case-analysis will honor the logic-constants pins/ports which have been bounded to Power/Ground separately.

When this variable is set to *false*, timing analysis will do traverse in PG networks, along with normal signal networks. The logic-constants pins/ports which have been bounded to Power/Ground will be ignored by case-analysis as well.

The default variable value is *false*.

See Also

- set_case_analysis
- report_timing

---

timing_early_launch_at_borrowing_latches

Removes clock latency pessimism from the launch times for paths that begin at the data pin of a transparent latch.

Data Types

Boolean

Default true

Description

When a latch is in its transparent phase, data arriving at the D-pin passes through the element as though it were combinational. To model this situation, whenever the IC Compiler tool determines that time borrowing is occurring at such a D-pin, it creates a timing path starting at the D-pin.

Sometimes there is a difference between the launch and capture latch latencies, due either to reconvergent paths in the clock network or different min and max delays of cells in the clock network. For setup paths, the tool uses the late value to launch and the early value to capture. This achieves the tightest constraint and prevents optimism. However,

Feedback

for paths starting from latch D-pins, this analysis is pessimistic because the data simply passes through and thus does not even "see" the clock edge at the latch.

When this timing variable is set to *true* (the default), such pessimism is eliminated by using the early latch latency to launch the path. Note that only paths that originate from a latch D-pin are affected. When the variable is set to *false*, late clock latency is used to launch all setup paths in the design.

When the *timing_early_launch_at_borrowing_latches* and *timing_remove_clock_reconvergence_pessimism* variables are both set to true, it is not possible to apply both pessimism removal techniques on the same timing path. Therefore, the tool applies early launch pessimism removal to paths that start from a transparent latch D-pin and applies CRPR to the remaining paths.

For a design with long clock paths, it might be preferable to set the *timing_early_launch_at_borrowing_latches* variable to *false*. Doing so allows CRPR to be applied to all paths, including those that start from a transparent latch D-pin. When clock paths are long, CRPR can be the more powerful pessimism reduction technique.

On the other hand, for a design with clock paths having long common segments, or where critical paths traverse several latches, leaving the *timing_early_launch_at_borrowing_latches* variable set to *true* might result in more pessimism removal overall, even though paths that start from a transparent D-pin do not get any CRPR credit.

See Also

- report_timing
- timing_remove_clock_reconvergence_pessimism

---

timing_edge_specific_source_latency

Controls whether or not the generated clock source latency computation considers edge relationship.

Data Types

Boolean

Default false

Description

When this variable is set to *true*, only the paths with the same sense relationship derived from generated clock definition are considered.

By default, all paths that fanout to a generated clock source pin are considered and the worst path is selected for generated clock source latency computation.

Feedback

See Also

- create_generated_clock

---

timing_enable_constraint_variation

Enables constraint variation in parametric on-chip variation (POCV) analysis.

Data Types

Boolean

Default false

Description

If you set this variable to *true* and set the *timing_pocvm_enable_analysis* variable to *true*, constraint variation is considered in parametric on-chip variation (POCV) analysis.

See Also

- get_timing_paths
- report_timing
- set_operating_conditions
- timing_pocvm_enable_analysis

---

timing_enable_cumulative_incremental_derate

Enables or disables the accumulation of incremental derate values.

Data Types

Boolean

Default false

Description

This variable controls whether the *set_timing_derate -increment* command replaces or adds to the existing incremental value stored at the specified object scope.

When set to *false* (the default), the *set_timing_derate -increment* command completely replaces any existing incremental derate setting stored at that object scope.

When set to *true*, the *set_timing_derate -increment* command numerically adds the specified incremental derate value to any existing incremental derate value stored at that object scope.

Feedback

This variable does not cause incremental derate settings across different object scopes (leaf cell/net, library cell, hierarchical cell, global) to be added together; such settings still follow the usual object scope precedence rules.

For example, given the following commands:

```
set_app_var timing_enable_cumulative_incremental_derate true
set_timing_derate -late -cell_delay -increment 0.01 [get_cells core/U123]
set_timing_derate -late -cell_delay -increment 0.01 [get_cells core/U123]
set_timing_derate -late -cell_delay -increment 0.04 [get_cells core]
set_timing_derate -late -cell_delay -increment 0.04 [get_cells core]
```

then the leaf cell incremental derate of 0.02 is used for core/U123. If the leaf cell incremental derate is subsequently removed:

```
reset_timing_derate -increment [get_cells core/U123]
```

then the hierarchical cell incremental derate of 0.08 is used for core/U123.

See Also

- set_timing_derate

---

timing_enable_multiple_clocks_per_reg

Enables analysis of multiple clocks that reach a single register.

Data Types

Boolean

Default true

Description

This variable enables analysis of multiple clocks that reach a register clock pin. When set to *true* (the default), all clocks reaching the register are analyzed simultaneously, including interactions between different clocks, such as data launch by one clock and data capture by another. If there are four or more clocks per register and the design contains level-sensitive registers, a high impact on runtime might occur.

If your design has a large number of different interacting clocks, you can set this variable to *false* to eliminate consideration of all clock interactions and get results more quickly. However, to get fully accurate results, leave the variable set to *true* and use the *set_false_path* command to explicitly specify the actual false interactions between mutually exclusive clocks.

Feedback

See Also

- check_timing
- create_clock
- create_generated_clock
- set_false_path

---

timing_enable_non_sequential_checks

Enables or disables library nonsequential checks in the design.

Data Types

Boolean

Default false

Description

This variable enables or disables analysis of library nonsequential checks in the design. The nonsequential arcs defined in the library will not be used for constraint checking unless this variable is set to *true*. This variable does not affect the data checks defined by the *set_data_check* command.

Enabling the nonsequential checks might cause delays if the signals reaching the related pin and the constrained pin do not belong to the same clock domain. Use the *set_multicycle_path* command to put appropriate constraints on such paths.

See Also

- printvar
- set_data_check
- set_multicycle_path

---

timing_enable_normalized_slack

Enables normalized slack analysis during timing updates.

Data Types

Boolean

Default false

Feedback

Description

When set to *true*, this variable enables normalized slack analysis during timing updates. When set to *false* (the default), this variable disables normalized slack analysis and reporting.

Normalized slack analysis is an optional analysis method that calculates normalized slack for each timing path:

```
normalized_slack = path_slack /
 idealized_allowed_propagation_delay_for_path
```

The tool computes the allowed propagation delay for the path using ideal clock edges; it ignores setup time, uncertainty, and clock latency. The allowed propagation delay can be a half-cycle, a full cycle, or multiple cycles. It can be more complicated to compute when the launch and capture clocks are different.

Normalized slack analysis can be used to determine the paths that limit the clock frequency. A normalized slack report prioritizes violating paths that are allowed few clock cycles. Fixing these paths first results in the most improvement in the clock period.

If normalized slack analysis is enabled during update timing, you can gather and report paths according to normalized slack by using the commands *report_timing -normalized_slack* and *get_timing_paths -normalized_slack*.

See Also

- printvar
- report_timing
- timing_max_normalization_cycles

---

timing_enable_slack_distribution

Determines whether to distribute slack evenly along time-borrowing paths.

Data Types

Boolean

Default false

Description

This variable determines whether to distribute slack evenly along a time-borrowing path containing transparent latches.

Feedback

By default, this variable is set to *false*, resulting in "greedy" time-borrowing behavior, in which latch time borrowing matches the D pin arrival time, producing zero slack at borrowing latches.

When this variable is set to *true*, during optimization, the tool sets the amount of time borrowing at transparent latches in a manner that distributes negative or positive slack evenly among the latches of a multistage latch path. The runtime and memory usage might be longer using this setting.

Slack distribution spreads all nonzero slack, negative or positive, across all multistage borrowing latch paths. By spreading positive path slacks, hold fixing and area QoR might be improved. This setting affects the behavior of the *compile_ultra* command in the Synopsys Physical Guidance (SPG) mode.

Slack distribution, when enabled, overrides automatic time borrowing (controlled by the *disable_auto_time_borrow* variable). Automatic time borrowing distributes only the worst negative slack across certain multistage latch paths.

See Also

- set_max_time_borrow
- disable_auto_time_borrow

---

timing_enable_through_paths

Enables or disables advanced analysis and reporting through transparent latches.

Data Types

Boolean

Default false

Description

When set to *true*, this variable enables advanced analysis through transparent latches during timing updates and reporting. When set to *false* (the default), this variable disables advanced analysis and reporting through transparent latches.

By default timer analyzes and reports paths through transparent latches as a series of path segments between latches. These segments can be reported together using the *report_timing* option *-trace_latch_borrow*. Max pin slacks (*max_rise_slack*, *max_fall_slack*) for a pin in the design can be affected by borrowing latches in the fanin of the pin, but are not affected by timing calculations in parts of the design past the first level of latches in the fanout of the pin.

With the advanced analysis through transparent latches, paths through latches can be reported as a single timing path. Pin slacks can be affected by timing calculations past

Feedback

the first level of latches in the fanout. In addition, specific paths through latches can be requested using the *-from*, *-through*, and *-to* options of *report_timing*, where the options specify objects that are separated by one or more transparent latches.

The advanced analysis is limited when there are latch loops in the design. The tool chooses specific latch data pins in the loops to act as loop breaker latches. For these latch data pins, the behavior is the same as if the variable *timing_enable_through_paths* was set to *false*. Reporting through these special latch data pins is not supported. The tool automatically selects which latch data pins to act as loop breaker latches. The user can guide the selection using the *set_latch_loop_breaker* command. Because of the runtime associated with the advanced analysis, by default the tool also selects some latch data pins outside loops to have the same behavior as if *timing_enable_through_paths* was *false*. The variable *timing_through_path_max_segments* can be used to control the selection of these pins.

To determine the current setting, use the following command:

```
printvar timing_enable_through_paths
```

See Also

- printvar
- report_timing
- timing_through_path_max_segments

---

timing_gclock_source_network_num_master_registers

Specifies the maximum allowed number of register clock pins clocked by the master clock in generated clock source latency paths.

Data Types

integer

Default 10000000

Description

This variable controls the maximum allowed number of register clock pins clocked by the master clock in generated clock source latency paths. The variable does not limit the number of registers 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.

Feedback

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 with a primary master clock that differs from the generated clock whose source latency paths are being computed.

timing_ignore_paths_within_block_abstraction

Causes timing paths entirely within a block abstraction to be ignored.

Data Types

Boolean

Default false

Description

The process of creating abstract blocks removes as much logic as possible from the internal portions of the block, while keeping cells and nets that are necessary to preserve paths that connect to the outside of the block. However, in a block abstraction, there can be residual paths entirely within the boundary logic that do not traverse outside the block.

Setting this variable to *true* causes the timing on paths that remain entirely within the abstracted block to be ignored. The tool behaves as if there is an implied false path constraint on such paths, both for reporting and for optimization. A path that starts and ends inside and abstracted block, but exits from and returns to the block somewhere in the middle, is not affected by this variable.

When this variable is *true*, it allows transparent interface optimization to focus on paths that traverse outside the abstracted blocks.

See Also

- create_block_abstraction

timing_input_port_default_clock

Determines whether a default clock is assumed at input ports for which a clock-specific input external delay is not defined.

Data Types

Boolean

Default false

Feedback

Description

This variable affects the behavior of the synthesis timing engine when timing a path from an input port with no clocked input external delay. When set to *true*, all such input ports are given 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. By default, no such imaginary clock is assumed.

See Also

- report_timing

---

timing_library_derate_is_scenario_specific

This variable determines whether the *set_timing_derate* and the *read_aocvm* commands apply to the current scenario or all scenarios for library cell objects.

Data Types

Boolean

Default false

Description

By default, the *set_timing_derate* command and the *read_aocvm* command, when applied to a library cell object, affect the timing of that cell in all scenarios, not just the scenario in which the command is executed. If you set this variable to true, the *set_timing_derate* and the *read_aocvm* commands affect only the scenario in which the commands are executed. The variable is set to false by default.

The variable setting has no effect on the *set_timing_derate* and *read_aocvm* commands applied to instances, in which case the *set_timing_derate* setting and the AOCVM tables are always scenario-specific.

Changing this variable at any time removes all timing derate settings and AOCVM tables already applied to library cell objects. To save runtime, set this variable early in the session, before you start using the *set_timing_derate* and *read_aocvm* commands.

This variable is not persistent and is not saved in the Milkyway design database.

Multicorner-Multimode Support

This variable determines whether the *set_timing_derate* command and the *read_aocvm* command apply to the current scenario or all scenarios for settings applied to library cell objects.

Feedback

Examples

The following example shows the default behavior. Here, for the *set_timing_derate* command, the previous setting "1.1" is overwritten by the later setting "1.2," even though they are set in two different scenarios. For the *read_aocvm* command, AOCVM tables for library cell objects are applied in Scenario1, even though they are loaded in Scenario2.

```
prompt> set_app_var timing_library_derate_is_scenario_specific false

prompt> current_scenario Scenario1
prompt> set_timing_derate -max -late 1.1  [get_lib_cells lib_pvt5/AN2]

prompt> current_scenario Scenario2
prompt> set_timing_derate -max -late 1.2  [get_lib_cells lib_pvt5/AN2]
prompt> read_aocvm lib_cell_table.txt

prompt> current_scenario Scenario1
prompt> report_aocvm

****************************************
Report : aocvm
Design : top
Scenario(s): Scenario1
****************************************

                  |        | Fully      | Partially  | Not        |
                  | Total  | annotated  | annotated  | annotated  |
 -----------------+--------+------------+------------+------------+
  Leaf cells      |     6  |          6 |          0 |          0 |
  Nets            |     7  |          0 |          0 |          7 |
 -----------------+--------+------------+------------+------------+
                  |    13  |          6 |          0 |          7 |


1

prompt> report_timing_derate -scenario [list Scenario1 Scenario2]

****************************************
Report : timing derate
Design : top
Scenario(s): Scenario1 Scenario2
 ...
****************************************

    Design    Derate                                  value     Scenario
------------------------------------------------------------------------


    Cell      Derate                                  value     Scenario
------------------------------------------------------------------------
```

```
    Lib Cell  Derate                                  value
-------------------------------------------------------------------------

    lib_pvt5/AN2
              clk_cell_delay_min_early                 -
              clk_cell_delay_min_late                  -
              clk_cell_delay_max_early                 -
              clk_cell_delay_max_late                1.20
              clk_cell_check_min_early                 -
              clk_cell_check_min_late                  -
              clk_cell_check_max_early                 -
              clk_cell_check_max_late                  -
              data_cell_delay_min_early                -
              data_cell_delay_min_late                 -
              data_cell_delay_max_early                -
              data_cell_delay_max_late               1.20
              data_cell_check_min_early                -
              data_cell_check_min_late                 -
              data_cell_check_max_early                -
              data_cell_check_max_late                 -
```

However, in the non-default state, with the variable set to true, each derate setting or the AOCVM table is specific to the scenario that is current when the constraint is applied:

```
prompt> set_app_var timing_library_derate_is_scenario_specific true
prompt> current_scenario Scenario1
prompt> set_timing_derate -max -late 1.1  [get_lib_cells lib_pvt5/AN2]

prompt> current_scenario Scenario2
prompt> set_timing_derate -max -late 1.2  [get_lib_cells lib_pvt5/AN2]
prompt> read_aocvm lib_cell_table.txt

prompt> current_scenario Scenario1
prompt> report_aocvm

****************************************
Report : aocvm
Design : top
Scenario(s): Scenario1
****************************************
No AOCVM derates.

1

prompt> report_timing_derate -scenario [list Scenario1 Scenario2]

****************************************
Report : timing derate
Design : top
Scenario(s): Scenario1 Scenario2
 ...
```

Feedback

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

    Design    Derate                                  value     Scenario
------------------------------------------------------------------------


    Cell      Derate                                  value     Scenario
------------------------------------------------------------------------


    Lib Cell  Derate                                  value     Scenario
------------------------------------------------------------------------

    lib_pvt5/AN2
              clk_cell_delay_min_early                 -        Scenario1
              clk_cell_delay_min_late                  -        Scenario1
              clk_cell_delay_max_early                 -        Scenario1
              clk_cell_delay_max_late                1.10       Scenario1
              clk_cell_check_min_early                 -        Scenario1
              clk_cell_check_min_late                  -        Scenario1
              clk_cell_check_max_early                 -        Scenario1
              clk_cell_check_max_late                  -        Scenario1
              data_cell_delay_min_early                -        Scenario1
              data_cell_delay_min_late                 -        Scenario1
              data_cell_delay_max_early                -        Scenario1
              data_cell_delay_max_late               1.10       Scenario1
              data_cell_check_min_early                -        Scenario1
              data_cell_check_min_late                 -        Scenario1
              data_cell_check_max_early                -        Scenario1
              data_cell_check_max_late                 -        Scenario1
    lib_pvt5/AN2
              clk_cell_delay_min_early                 -        Scenario2
              clk_cell_delay_min_late                  -        Scenario2
              clk_cell_delay_max_early                 -        Scenario2
              clk_cell_delay_max_late                1.20       Scenario2
              clk_cell_check_min_early                 -        Scenario2
              clk_cell_check_min_late                  -        Scenario2
              clk_cell_check_max_early                 -        Scenario2
              clk_cell_check_max_late                  -        Scenario2
              data_cell_delay_min_early                -        Scenario2
              data_cell_delay_min_late                 -        Scenario2
              data_cell_delay_max_early                -        Scenario2
              data_cell_delay_max_late               1.20       Scenario2
              data_cell_check_min_early                -        Scenario2
              data_cell_check_min_late                 -        Scenario2
              data_cell_check_max_early                -        Scenario2
              data_cell_check_max_late                 -        Scenario2
```

Feedback

See Also

- set_timing_derate
- report_timing_derate
- set_min_library

---

timing_library_max_cap_from_lookup_table

Determines whether the tool should use the operating frequency to determine the maximum pin load.

Data Types

Boolean

Default false

Description

This variable specifies whether the tool should use the operating frequency to determine the maximum pin load. The tool uses a single value for the maximum pin load by default.

When this variable is set to true, the tool uses the operating frequency to determine the maximum pin load.

To see the current value of this variable, use the *printvar timing_library_max_cap_from_lookup_table* command.

See Also

- set_max_capacitance
- report_constraint

---

timing_max_normalization_cycles

Sets an upper limit for the denominator when calculating normalized slack.

Data Types

integer

Default 4

Feedback

Description

Normalized slack analysis is an optional analysis method that calculates normalized slack for each timing path:

```
normalized_slack = path_slack /
 idealized_allowed_propagation_delay_for_path
```

The tool computes the allowed propagation delay for the path using ideal clock edges; it ignores setup time, uncertainty, and clock latency. The allowed propagation delay can be a half-cycle, a full cycle, or multiple cycles. It can be more complicated to compute when the launch and capture clocks are different.

The *timing_max_normalization_cycles* variable limits the runtime and memory used in this analysis by placing an upper limit on the denominator of the fraction used to calculate the normalized slack. The maximum allowed value of this denominator is this variable setting multiplied by the period of the launch clock for the path.

Setting a larger value supports a larger range of allowed propagation delays, possibly at the cost of more runtime and memory usage.

See Also

- printvar
- report_timing
- timing_enable_normalized_slack

---

timing_max_parallel_computations

Sets the maximum degree of parallelism used for parallel timing updates.

Data Types

integer

Default 0

Group

timing

Description

This application variable specifies the degree of parallelism used for parallel timing updates. Increasing the parallelism results in reduced runtime at the expense of memory.

Feedback

When this variable has a value of *0* (the default), the number of cores used for timing update is specified by the value of the *-max_cores* option of the *set_host_options* command.

When this variable has a value of *1*, parallel timing update is disabled.

When this variable has a value that is larger than 1 but smaller than the value of the *set_host_options -max_cores* option, the number of cores being used must not exceed this limit and a smaller memory footprint is achieved.

When this variable has a value that is larger than the value of the *set_host_options -max_cores* option, the number of cores being used must not exceed the limit set with the *set_host_options* command and better runtime speedup can be achieved.

The actual degree of parallelism (number of processes/threads) must not exceed the value specified in this variable, but will not always be exactly the same. The tool derives the number internally to achieve the best performance.

See Also

- set_host_options

---

timing_ocvm_precedence_compatibility

Specifies whether to fall back to conventional on-chip variation (OCV) derating while advanced on-chip variation (AOCV) or parametric on-chip variation (POCV) is enabled.

Data Types

Boolean

Default false

Description

When this variable is set to false (the default), for each object, both OCV and AOCV or POCV derate settings are considered, and the more specific setting is used. This is the default order of usage, from highest to lowest priority:

```
i.  OCV leaf cell derate setting
   ii.  AOCV or POCV lib-cell derate setting
  iii.  OCV lib-cell derate setting
   iv.  AOCV or POCV hier-cell derate setting
    v.  OCV hier-cell derate setting
   vi.  AOCV or POCV design derate setting
  vii.  OCV design derate
```

Feedback

When this variable is set to true, the OCV derate settings are always ignored for AOCV or POCV analysis, resulting in the following order of usage, from highest to lowest priority:

```
i.  AOCV or POCV lib-cell derate setting
   ii.  AOCV or POCV hier-cell derate setting
  iii.  AOCV or POCV design derate setting
```

This variable has an effect only when the *timing_pocvm_enable_analysis* variable is set to true.

See Also

- read_ocvm
- remove_ocvm
- report_ocvm
- timing_pocvm_enable_analysis

---

timing_pocvm_corner_sigma

Specifies the standard deviation used to calculate worst-case values from statistical parameters during parametric on-chip variation analysis.

Data Types

float

Default 3

Description

Parametric on-chip variation (POCV) analysis internally computes arrival times, required times, and slack values based on statistical distributions. When performing comparisons between these statistical quantities, the tool needs to know what values in the distribution are considered worst-case.

The default behavior is to choose values at 3.0 standard deviations away from the mean value. In other words, for an arrival time distribution, the worst-case early arrival is 3.0 standard deviations below the mean, and the worst-case late arrival is 3.0 standard deviations above the mean.

You can use this variable set a different number of standard deviations away from the mean to determine the worst-case values for timing reports. Use a lower value such as 2.5 for less worst-case variation and more relaxed timing constraints. Conversely, use a higher value such as 3.5 for more worst-case variation and more restrictive timing constraints.

Feedback

See Also

- timing_pocvm_enable_analysis

---

timing_pocvm_enable_analysis

Enables parametric on-chip variation (POCV) timing analysis.

Data Types

Boolean

Default false

Description

When this variable is set to *true*, the tool performs parametric on-chip variation (POCV) timing analysis. In this analysis mode, the tool internally computes arrival times, required times, and slack values as statistical distributions rather than fixed minimum and maximum values.

To determine the cumulative delay of a path, the POCV mode statistically combines the delay distribution of each stage. This is more accurate than simply adding the worst-case value from each stage. The resulting delay and slack values are more realistic and less pessimistic than values calculated by ordinary addition.

Before you can use POCV analysis, you must specify the statistical delay behavior of the logic gates used in the design and read that information into the tool using the *read_ocvm* command.

Note that POCV timing analysis uses more runtime than conventional timing analysis.

See Also

- read_ocvm
- report_timing
- timing_pocvm_corner_sigma

---

timing_pocvm_precedence

Controls the precedence of how multiple POCV tables that are file-based or library-based are applied to a timing arc

Data Types

string

Feedback

Default file

Description

Set this app-option to one of these values:

o file (the default) - POCV coefficient file takes precedence over LVF data available in library.

o library - POCV LVF data available in library takes precedence over coefficient file.

o lib_cell_in_file - the tool uses the following priority, in decreasing order of precedence, to determine the table that applies to the arc:

```
o Library cell table
 o Hierarchical cell table
 o Design table
```

This app-option is effective only when the option *timing_pocvm_enable_analysis* is set to *true*.

See Also

- timing_pocvm_enable_analysis

---

timing_remove_clock_reconvergence_pessimism

Enables or disables clock reconvergence pessimism removal.

Data Types

Boolean

Default false

Description

This variable, when set to *true*, instructs the synthesis timing engine to remove clock reconvergence pessimism from slack calculation and minimum pulse width checks.

Clock reconvergence pessimism (CRP) is a difference in delay along the common part of the launching and capturing clock paths. The most common causes of CRP are reconvergent paths in the clock network and different minimum and maximum delay of cells in the clock network.

Any effective change in the value of the *timing_remove_clock_reconvergence_pessimism* variable causes a full *update_timing*. You cannot perform one *report_timing* operation that considers CRP and one that does not without a full *update_timing* in between.

To run optimization with CRP removal, the clock network must be set as dont_touch:

```
prompt> set_app_var timing_remove_clock_reconvergence_pessimism true
true

prompt> report_timing
```

See Also

- report_timing
- timing_crpr_threshold_ps
- timing_clock_reconvergence_pessimism

---

timing_report_attributes

Specifies the list of attributes reported by the *report_timing -attributes* command.

Data Types

list

Default {dont_touch dont_use map_only size_only ideal_net infeasible_paths}

Description

This variable specifies the attributes reported by the *report_timing -attributes* command. The list can contain any of the following attributes: *dont_touch*, *dont_use*, *map_only*, *size_only*, *infeasible_paths* (Design Compiler and DC Explorer only), *ideal_network*, *clock_marking*, *clock_marking*, *pass_gate*, *always_on*, *net_pattern*, and *ideal_net*.

When you use the *-attributes* option of the *report_timing* command, the report shows the attributes that apply to each path increment under the heading "Attributes". A symbol key at the beginning of the path report lists the letter codes used for the attributes being reported. For example,

```
Attributes:
    d - dont_touch
    u - dont_use
   mo - map_only
   so - size_only
    i - ideal_net or ideal_network
  inf - infeasible path
```

To determine the current value of this variable, type *printvar timing_report_attributes*. For a list of all timing variables and their current values, type *print_variable_group timing*.

Feedback

See Also

- compile
- report_timing

---

timing_report_fast_mode

Enables the fast *report_timing* mode, which reports the worst timing paths among all path groups instead of individual path groups.

Data Types

Boolean

Default false

Description

This variable, when set to true, selects the fast reporting mode for the *report_timing* and *get_timing_paths* commands.

When this variable is set to false (the default), the worst timing paths are reported separately for each path group. When this variable is set to true, the worst paths in the design are reported, irrespective of path groups, so fewer paths are reported.

Setting this variable to true results in the following changes in reporting behavior:

- The values specified with the *-max_paths* and *-nworst* options apply to all paths in the design rather than paths in each path group. For example, if the design has 12 path groups and *-max_paths* is set to 3, only 3 paths are reported instead of 36.
- The paths reported by the command are ordered strictly by slack and are not separated by path group.
- When the *-max_paths* and *-nworst* options are set to values larger than 1, only paths with negative slack are reported; paths with positive slack are omitted from the report.

Setting this variable to true results in the same reporting behavior as PrimeTime with respect to path groups and the *-max_paths* and *-nworst* options.

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

See Also

- get_timing_paths
- report_timing

Feedback

timing_report_union_tns

Specifies whether to use the union method for reporting total negative slack and number of violating endpoints.

Data Types

Boolean

Default true

Description

When this variable is set to *true* (the default), the tool counts a violating endpoint only once when computing the total negative slack (TNS) and the number of violating endpoints for a path group, scenario, or design. The TNS for a path group is the sum of worst violations for all endpoints in that path group. The TNS for a scenario is the sum of worst violations in that scenario for all endpoints, irrespective of the path group. The TNS for a design is the sum of worst violations for all endpoints, irrespective of path group or scenario.

When this variable is set to *false*, the tool computes the scenario TNS as the sum of TNS for all path groups in that scenario. The multi-scenario TNS is computed as the sum of TNS for all scenarios.

See Also

- report_qor

timing_save_library_derate

Controls whether the library cell derate settings are saved with the design (e.g., in milkyway cell or ddc files).

Data Types

Boolean

Default false

Description

This variable controls whether the library cell derate settings are saved with the design.

By default (false), these settings are not saved with the design, and need to be re-applied after re-opening the design in IC Compiler or Design Compiler. To save these settings with the design, set this variable to true before saving the design in IC Compiler or Design Compiler.

Feedback

See Also

- read_file
- write
- set_timing_derate

---

timing_scgc_override_library_setup_hold

Specifies whether you can override library-specified clock-gating setup and hold values with new values specified with the *set_clock_gating_check* command.

Data Types

Boolean

Default true

Group

timing

Description

When this variable is set to *true* (the default), you can use the *set_clock_gating_check* command to specify clock-gating setup and hold values on cells or pins, overriding the library-specified values on those cells or pins.

When this variable is set to *false*, the *set_clock_gating_check* command cannot override library-specified clock-gating setup and hold values on cells or pins.

Note that setup and hold values specified with the *set_clock_gating_check* command, when specified for clocks or the design, always have lower priority than library-specified values, irrespective of this variable setting. This variable only affects values set on cells or pins.

When the variable is set to *true* (the default), you can use the *set_clock_gating_check* command to set the setup and hold times to 0 for clock-gating checks at the design level, which enables automatic clock-gating check inferring and does not override any library values.

See Also

- set_clock_gating_check
- report_clock_gating

Feedback

- report_clock_gating_check
- report_timing

---

timing_self_loops_no_skew

Affects the behavior, runtime, and CPU usage of *report_timing* and *compile*.

Note: This variable will be obsolete in the next release. Please adjust your scripts accordingly.

Data Types

Boolean

Default false

Description

Affects the behavior, runtime, and CPU usage of *report_timing* and *compile*. When set to *true*, clock skew is eliminated for a path that starts and ends at the same register. When set to *false* (the default value), clock skew is not eliminated. Thus, the timing for such paths is pessimistic. To obtain the more accurate behavior of no clock skew (uncertainty) for such paths, set this variable to *true*. However, note that runtime and memory usage might increase significantly.

To determine the current value of this variable, type *printvar timing_self_loops_no_skew*. For a list of all *timing* variables and their current values, type *print_variable_group timing*.

See Also

- compile
- report_timing

---

timing_separate_clock_gating_group

Specifies if a separate cost group is used for clock-gating checks in timing analysis, reports, and optimization.

Data Types

Boolean

Default false

Group

timing

Feedback

Description

When this variable is set to *true*, a separate cost group named ***clock_gating_default*** is created for all clock-gating checks.

When set to *false* (the default), the clock-gating check is applied to the cost group of the clock being gated.

If multiple scenarios exist, a cost group is created for clock-gating checks for each scenario when this variable is *true*.

You can change the weight and critical range settings for this cost group using the *group_path* command with the *-name ***clock_gating_default**** option.

See Also

- get_path_groups
- group_path
- report_clock_gating
- report_clock_gating_check
- report_constraint
- report_path_group
- report_qor
- report_timing

---

timing_show_net_in_timing_loop

Shows net names along with pins in timing loops

Data Types

Boolean

Default false

Description

When this variable is set to *true*, net names are reported with pins in timing loops report using *update_timing* command. When this variable is set to *false* (default value), it reports the only the pin names in the timing loops report which is the default behaviour.

Feedback

See Also

- update_timing
- check_timing
- report_timing

---

timing_through_path_max_segments

Controls the maximum number of latches for reporting paths through latches.

Data Types

Integer

Default 5

Description

When *timing_through_path_max_segments* is set to a non-zero value, the tool selects some latch data pins on paths with many transparent latches to behave as they would if *timing_enable_through_paths* were *false*, limiting the reporting on the long path but speeding up analysis. With a small non-zero value, many transparent latch data pins in the design will be selected. With a larger value, fewer will be selected.

When *timing_through_path_max_segments* is set to 0, no latch data pins are selected. Reporting on paths through many latches is allowed, but analysis may be slower.

The variable *timing_through_path_max_segments* has no effect if the variable *timing_enable_through_paths* is set to *false*.

To determine the current setting, use the following command:

```
printvar timing_through_path_max_segments
```

See Also

- printvar
- report_timing
- timing_enable_through_paths

---

timing_use_ceff_for_drc

Specifies whether to use effective capacitance (Ceff) or total capacitance (Ctot) for DRC fixing of maximum capacitance violations.

Feedback

Data Types

Boolean

Default false

Description

When this variable set to *false* (the default), total capacitance (Ctot) values are compared against the maximum capacitance limit during DRC fixing. When this variable is set to *true*, effective capacitance (Ceff) values are used instead.

Total capacitance is the sum of all the capacitance values in an RC network. Effective capacitance is a lumped value that results in a similar delay effect as the full RC network, calculated as the average capacitive loading that a gate observes at the delay trip-point.

The effective capacitance can be very different from the maximum total capacitive loading experienced by the gate. Choosing to use effective capacitance can possibly result in extrapolation errors.

Setting the variable to *true* may result in less pessimistic DRC fixing. Before you set this variable to *true*, be sure to carefully evaluate your capacitance fixing and signoff methodology.

See Also

- report_delay_calculation
- set_max_capacitance

---

timing_use_clock_specific_transition

Propagate the transition from the specific clock path for latency calculation.

Data Types

Boolean

Default true

Description

When this variable is set to *true* (the default value), the tool only propagates the transition of the clock path, for the purpose of clock latency calculation. If there are multi-input gates on the clock network, the transition of non-clock inputs are ignored during clock latency calculation.

For generated clocks defined on an output of a gate, the tool propagates the transition from the path connected to its master clock, and uses that transition value for the purpose of calculating the clock latency for the generated clock.

Feedback

When set to *false*, the tool allow transition from the non-clock input of multi-input gates along the clock path to be used for clock latency calculation.

In addition, when set to *false*, the generated clock defined at the output of a gate uses zero transition at the generated clock source.

timing_use_driver_arc_transition_at_clock_source

Uses the backward cell arc to compute a realistic driver model at the driver pin for primary clock sources and also a generated clock that cannot trace back to its master clock.

Data Types

Boolean

Default true

Description

When set to *true* (the default value), the tool uses the backward cell arcs, (when at least one exists), to compute a worst-case driver model. This behavior applies to the primary clock sources, which are defined by the *create_clock* command, and generated clock sources (defined by the *create_generated_clock* command) that cannot trace back to its master clock.

When set to *false* the tool asserts a zero-transition, ideal ramp model at the driver pin.

See Also

- create_clock
- create_generated_clock

timing_use_enhanced_capacitance_modeling

Specifies whether to use the enhanced capacitance modeling information available in the library description of a pin.

Data Types

Boolean

Default true

Description

When this variable is set to *true* (the default), the tool uses the enhanced capacitance modeling information available in the library description of a pin, consisting of different capacitance values for rise and fall transitions, and possibly a range of worst-case values

Feedback

for each type of transition. This setting is recommended for better timing correlation with the PrimeTime tool.

When this variable is set to *false*, the tool uses the single default capacitance value specified in the library description of the pin and ignores any enhanced capacitance information.

The library description of pin in Liberty format can contain both a default capacitance value and more specific values or ranges of capacitance values for rise and fall transitions. For example,

```
pin("IN1") {
  direction : input;
  capacitance : 0.0067542;
  rise_capacitance : 0.0123321;
  fall_capacitance : 0.0056745;
  rise_capacitance_range (0.0045670, 0.012345);
  fall_capacitance_range (0.005656, 0.0123123);
}
```

When the variable is set to *true*, the tool uses the worst-case capacitance values specified by the rise and fall range attributes (for example, the high value for late arrival timing and the low value for early arrival timing). When the variable is set to *false*, the tool uses the single value specified by the plain "capacitance" attribute and ignores the rise and fall capacitance attributes.

See Also

- set_operating_conditions

---

timing_using_default_clock_gated_check

Specifies whether to use default clock gated check setup to align the behaviour with PT/FC.

Data Types

Boolean

Default false

Description

When this variable is set to *true*, the tool uses the default clock gated check setup like PT/FC .

Feedback

tio_allow_mim_optimization

Enables support for multiply instantiated modules (MIMs) during interface optimization performed using the *route_opt* and *focal_opt* commands.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

false in DC Explorer

Description

This variable enables support for multiply instantiated modules (MIMs) during interface optimization performed using the *route_opt* and *focal_opt* commands.

When interface optimization is enabled on MIMs,

1. The tool optimizes inside MIMs.
2. The tool ensures that optimization done in any instance improves timing across all the instances of the MIM.
3. The tool replicates the netlist changes across all the instances.
4. At the end of optimization, the reference cell is taken through the block update steps consisting of the *legalize_placement*, *route_zrt_eco*, and *create_block_abstraction* commands.

When this variable is set to *false*, the tool cannot optimize the interface of blocks that are multiply instantiated.

See Also

- set_top_implementation_options

tio_preload_block_site_rows

Specifies the IC Compiler to load sub-block site rows during initial linking in flows involving transparent interface optimization (TIO). This variable is to be used for designs with different site array configurations in top and sub-blocks.

Data Types

Boolean

Feedback

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Setting this variable as *true* makes IC Compiler to load the sub-block site rows during initial linking of top-level design. This is useful for flows that need to process site rows and site types of both Top-level and sub-blocks. An example usage is when running transparent interface optimization (TIO) on designs having different site rows in top and sub-blocks.

To use this feature, you must set the variable to *true* in the IC Compiler tool before you open the design.

See Also

- set_top_implementation_options

---

tio_preserve_routes_for_block

Setting this variable to true enables zroute based route preservation inside the blocks when transparent interface optimization is performed using the route_opt or focal_opt commands.

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

false in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

During transparent interface optimization, when the tool changes a cell inside a block abstraction, the net identities can also change. For example, when inserting a buffer creates a new net, although the tool seeks to place the new cells along the route and maintain route topology, it does not simultaneously reassign the net identity of the route metallization geometries associated with the net. Route preservation performs this task in a process executed after the optimization phase finishes and before ECO routing begins. The presence of the route segments bound to the nets assist the ECO router to find a similar solution to the one obtained before optimization. In this way, preserving the routes

Feedback

improves correlation of the routing before and after optimization and therefore acts to improve the convergence in timing before and after ECO.

This variable is only active when Zroute is selected. If Zroute is disabled by using the set_route_mode_options -zroute false command, this variable is not active.

tp_analyze_false_path

analyze false path constraints in test points

Data Types

Boolean

Default false

Description

analyze false path constraints in test points

Examples

A typical use mode will consist on this sequence of commands:

```
prompt> set  tp_analyze_false_path  "true"

prompt> set_testability_configuartion -target {multicycle_paths}
 -clock_signal clk -control_signal TM

prompt> run_test_point_analysis

prompt> preview_dft

prompt> insert_dft
```

u

ungroup_keep_original_design

Controls whether the *ungroup* and *compile* commands keep the original design when a design is ungrouped.

Data Types

Boolean

Default false

Feedback

Group

none

Description

By default, the *ungroup* and *compile* commands delete the original design if all the instances of a design have been ungrouped and there are no other references to the design.

When set to *true*, the original design is preserved.

For example, assume there are two instances of the *mid* design named *mid1* and *mid2*. If you run the *ungroup -flatten -all* command, after the tool collapses the hierarchies, the design called *mid* is deleted from memory if there are no other references to the design. This variable is used to change the behavior. When this variable is set to *true*, the ungrouped design is preserved.

See Also

- compile
- set_ungroup
- ungroup

---

uniquify_keep_original_design

Controls the *uniquify* command to keep the original design when a multiply-instantiated design is uniquified.

Data Types

Boolean

Default false

Group

none

Description

By default, the *uniquify* command deletes the original design if all the instances of a design have been uniquified and there are no other references to the design from anywhere else.

When set to *true*, the original design is preserved.

Feedback

For example, if there are two instances of the *mid* design, uniquify creates two new designs named *mid_1* and *mid_2*. By default the original design named *mid* is deleted. This variable is used to change this behavior.

See Also

- uniquify

---

uniquify_naming_style

Specifies the naming convention to be used by the *uniquify* command.

Data Types

string

Default %s_%d

Description

This variable specifies the naming convention to be used by the *uniquify* command. The variable string must contain only one %s (percent s) and one %d (percent d) character sequence. To use a percent sign in the design name, two are needed in the string (%%).

See Also

- uniquify

---

upf_add_power_state_21_syntax

Specifies whether the tool uses the UPF pre-2.1 style or UPF 2.1 (and beyond) syntax for the *add_power_state* command.

Data Types

Boolean

Default false

Group

none

Description

Setting the variable to *true* enforces the UPF 2.1 (and beyond) syntax of the *add_power_state* command. Setting the variable to *false* enables the UPF pre-2.1 syntax. The following examples show the different syntax styles.

Feedback

UPF pre-2.1 syntax: add_power_state SS -state SS_1 {-supply_expr {power == {FULL_ON 0.9} && ground == {FULL_ON 0.0}}}

UPF 2.1 syntax: add_power_state SS -state {SS_1 -supply_expr {power == {FULL_ON 0.9} && ground == {FULL_ON 0.0}}}

After you set this variable, it is marked read-only. The Design Compiler tool does not allow you to mix the syntax styles of the *add_power_state* command. The *add_power_state* commands in the UPF can only be in one style or the other. Choose the style by setting the variable before loading the UPF, but you can only do this one time for a Design Compiler session. This restriction of ensures that the syntax style is fixed for a session.

See Also

- add_power_state

---

upf_allow_DD_primary_with_supply_sets

Allows using domain dependent supply nets of the design as the primary supply of the power domain when supply sets are used in the design.

Data Types

Boolean

Default false

Description

When supply sets are used in a design, domain dependent supply nets cannot be used as the primary supply of the power domain. Setting this variable to *true* removes this restriction. The default value of this variable is *false*.

After you read the UPF file, the tool ignores any changes to this variable setting. To change the variable setting, remove the UPF file, change the value of the variable, and reload the UPF file.

See Also

- load_upf
- remove_upf

---

upf_allow_is_isolated_output_check

Controls if is_isolated attribute is to be ignored on output pins.

Feedback

Data Types

Boolean

Default true

Description

This variable controls whether is_isolated attribute should be ignored on output pins. The default value of this variable is *true*.

When this variable is set to *true* (the default), tool checks if all load pins are equal or more always on than the driver pin irrespective of whether this pin has is_isolated attribute or not. If this check fails, MV-514 is issued by the tool. To skip the on_ness check with receiver for an is_isolated output pin, set this variable to false.

---

upf_allow_iso_on_dont_touch_nets

Controls the precedence between isolation strategy and dont_touch attribute of nets.

Data Types

Boolean

Default true

Group

mv

Description

This variable, when set to false, gives precedence to dont_touch attribute on nets over isolation strategy; i.e., isolation cell insertion will not happen on user specified dont_touch nets.

The default value is *true*, in which case isolation strategy is given precedence and isolation cells are inserted on dont_touch nets with a warning.

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

```
prompt> printvar upf_allow_iso_on_dont_touch_nets
```

See Also

- set_isolation

Feedback

upf_allow_ls_on_dont_touch_nets

Controls the precedence between level shifter cell insertion and dont_touch attribute of nets.

Data Types

Boolean

Default false

Group

mv

Description

This variable, when set to *false*, gives precedence to dont_touch attribute on nets over level shifter cell insertion; i.e., level shifter cell insertion will not happen on user specified dont_touch nets.

When set to *true*, in which case level shifter cell insertion is given precedence and level shifter cells will be inserted on dont_touch nets.

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

```
prompt> printvar upf_allow_ls_on_dont_touch_nets
```

See Also

- set_level_shifter

upf_allow_non_bias_domain_in_bias_scope

Allows the use of non-bias power domains in a bias enabled scope.

Data Types

Boolean

Default false

Description

This option enables the use of non-bias power domains in bias enabled scopes. By default the tool does not allow the use of non-bias power domains in a bias scope. This app-option can be set to true to allow non-bias power domains to be defined within scopes where enable_bias is set to true/derived.

Feedback

See Also

- set_design_attributes

---

upf_allow_or_operator_in_add_power_state_supply_expr

Variable control to allow OR operator in -supply_expr of add_power_state command.

Data Types

Boolean

Default false

Description

This variable controls whether tool allows logical OR operator '||' in supply expression of add_power_state commands. The default value of this variable is *false*.

This variable must be set before loading the UPF or before defining any supply set states or group states. Once this variable is set, its value cannot be modified.

When this variable is true, || will be accepted in add_power_state -supply_expr. When this variable is not set to true, presence of || in supply_expr will result in an error.

Example:

```
prompt> set upf_allow_or_operator_in_add_power_state_supply_expr truei
prompt> add_power_state SSM -state SSM1 {-supply_expr {(power ==
 `{FULL_ON, 1.0} || power == `{FULL_ON, 0.9}) && ground == `{FULL_ON,
 0.0}}}
1
```

Only flat flow is supported with this variable. Hierarchical flow is not supported. characterize will result in incorrect system PST. propagate_constraints will not propagate all the states correctly.

See Also

- add_power_state

---

upf_allow_power_gating_cell_for_retention

Enables *power_gating_cell* attribute to be considered as valid attribute for identifying library cells as retention cells. This variable can be used only in UPF mode and in dc_shell. It should be defined before RTL is read.

Feedback

Data Types

Boolean

Default false

Group

power

Description

The *upf_allow_power_gating_cell_for_retention* variable enables and disables the *power_gating_cell* attribute for identifying retention cells in libraries.

When set to *true power_gating_cell* attribute will be considered as a valid attribute for identifying retention cells.

When set to *false power_gating_cell* attribute will not be considered as a valid attribute instead only *retention_cell* attribute will be considered as a valid attribute for identifying retention cells.

Variable will be deprecated in next major release.

---

upf_allow_rbm_in_upf_prime

Enables rule based matching in UPF Prime flow when set to true.

Data Types

Boolean

Default false

Description

Specifies whether rule based matching will be allowed while loading UPF during UPF Prime flow. This only suports the default Golden UPF query rules, so any options given with set_query_rules will be ignored. This option is ignored when running in Golden UPF flow.

See Also

- set_query_rules
- enable_golden_upf

---

upf_allow_refer_before_define

Allows UPF commands to refer to undefined UPF objects in the nested scope.

Feedback

Data Types

Boolean

Default false

Group

none

Description

When loading the top-only UPF, some UPF commands refer to block-level UPF objects which are not defined in top-only UPF. By default, it is not allowed to refer these undefined UPF objects. These undefined objects must be defined in the block-level UPF.

Examples

In the following example, assume i_sub/pd_sub is not defined yet:

```
prompt> associate_supply_set ss -handle i_sub/pd_sub.primary
Error: Supply set 'i_sub/pd_sub.primary' does not exist. (UPF-163)
0
```

When this variable is set to *true*, the reference of undefined block-level UPF objects are allowed.

```
prompt> set upf_allow_refer_before_define true
true
prompt> associate_supply_set ss -handle i_sub/pd_sub.primary
1
```

The i_sub/pd_sub.primary object is subsequently defined in the block-level UPF.

See Also

- load_upf
- save_upf

---

upf_ao_cells_without_primary_pg_pin

Option to turn on support of always-on library cells with no primary power pin

Data Types

Boolean

Default false

Feedback

Group

mv

Description

When this variable is set to *true*, tool will only use always-on buffer, inverter, TIE and isolation library cells with no primary power pin whenever regular dual-rail library cells are required.

See Also

- insert_mv_cells

---

upf_apply_retention_attribute_on_non_retention_macro

Allows specifying non-Retention macros in element list of *set_retention* and *set_retention_elements* commands.

Data Types

Boolean

Default false

Group

upf

Description

When this variable is set to true, non-Retention macros will be allowed in element list of *set_retention* and *set_retention_elements* commands. All the Retention attributes will be set on them.

Existing behavior of Design Compiler not mapping non-retention macro cells to retention macro cells will continue to be the same. It is user's responsibility to replace the non-retention macro cells with their retention equivalent. Failure to do so will reflect in check_mv_design via UPF-562 and UPF-563 messages.

Examples

Consider input design has non-Retention macro (MACRO_1) in power domain PDT:

```
prompt> set_retention RET_PDT -domain PDT -elements { MACRO_1 }
Warning: The cell MACRO_1 specified with the -elements option of the
 set_retention
or map_retention_cell command is neither sequential nor hierarchical.
 (UPF-146)
0
prompt> set upf_apply_retention_attribute_on_non_retention_macro true
```

Feedback

```
prompt> set_retention RET_PDT -domain PDT -elements { MACRO_1 }
1
```

See Also

- set_retention
- set_retention_elements

---

upf_auto_iso_clamp_value

Sets the clamp value for isolation strategies inferred by the DC Explorer tool.

Data Types

string

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

0 in DC Explorer

Group

none

Description

This variable sets the clamp value of isolation strategies inferred by the DC Explorer tool. It can be set to either *0* or *1*.

See Also

- set_isolation
- set_isolation_control

---

upf_auto_iso_enable_source

Sets the source of the isolation control signal for isolation strategies inferred by the DC Explorer tool.

Data Types

string

Feedback

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

root_cell in DC Explorer

Group

none

Description

This variable determines the source of the isolation control signal for isolation strategies inferred by the DC Explorer tool.

Leave this variable set to *root_cell* (the default) if control signal comes from a power controller block instantiated in the design root scope.

Set this variable to *top_level_port* if the control signal comes from a top-level port.

See Also

- set_isolation
- set_isolation_control

---

upf_auto_iso_isolation_sense

Sets the isolation sense for isolation strategies inferred by the DC Explorer tool.

Data Types

string

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

low in DC Explorer

Group

none

Description

This variable sets the logical sense of the isolation control signal for isolation strategies inferred by the DC Explorer tool. It can be set to either *low* or *high*.

Feedback

See Also

- set_isolation
- set_isolation_control

---

upf_block_partition

Specifies a list of hierarchical cells to be treated as block-level designs; their UPF is not written out by the *save_upf* command.

Data Types

list

Default ""

Description

Use this variable to specify a list of cells to be treated as block-level hierarchies. When you save the top-level UPF using the *save_upf* command, the tool excludes the UPF associated with these blocks.

This variable is associated with the current design. The specified hierarchical cells must exist in the design when you set the variable. If you change to a different design, you need to set the variable again.

See Also

- save_upf
- check_upf

---

upf_charz_allow_port_punch

Allows UPF-related port punching to occur within the *characterize* command on the blocks being characterized.

Data Types

Boolean

Default true

Group

mv

Feedback

Description

The *upf_charz_allow_port_punch* variable, when set to *true*, allows the *characterize* command to modify the block interfaces to automatically port punch the control signals of the power management cells from the top design and bring them into the block.

See Also

- characterize

---

upf_charz_create_compact_pst

Controls whether characterize should create a compact derived PST for the block being characterized.

Data Types

Boolean

Description

When this variable is set to *false* (the default), characterize will create a derived group and/or a derived PST for the block being characterized. This derived group/PST will be non-compact in that it will have all the states expanded.

When this variable is set to *true*, characterize will try to create a compact derived PST. This compact derived PST will use "*" annotation for cases where all the states of a supply are considered in the derived PST. As all the states are not explicitly listed in the derived PST, this PST will be compact.

Example: Consider the following full-chip UPF -

```
>>>>
set_design_attributes -elements {.} -attribute
 enable_state_propagation_in_add_power_state true

create_supply_set SST
create_power_domain PDTOP -supply {primary SST}
add_power_state SST -state S1 {-supply_expr {power == {FULL_ON 0.9}}}
add_power_state SST -state S2 {-supply_expr {ground == {FULL_ON 0.0}}}

create_supply_set SSM
create_power_domain PDMID -supply {primary SSM} -elements {mid_inst}
add_power_state SSM -state S3 {-supply_expr {power == {FULL_ON 0.9}}}
add_power_state SSM -state S4 {-supply_expr {power == {FULL_ON 1.0}}}
add_power_state SSM -state S5 {-supply_expr {ground == {FULL_ON 0.0}}}
add_power_state SSM -state S6 {-supply_expr {ground == {OFF}}}

create_pst PST_TOP -supplies {SST.power SST.ground SSM.power SSM.ground}
add_pst_state PST_TOP_S1 -pst PST_TOP -state {S1 S2 S3 S5}
```

```
add_pst_state PST_TOP_S2 -pst PST_TOP -state {S1 S2 S4 S5}
>>>>
```

When this variable is set to *false* (the default), below is how the characterized UPF for the block "mid_inst" will look like. As can be seen, the derived PST has two rows with the names of all the states listed.

```
>>>>
create_pst pst -supplies {SST.power SST.ground SSM.power SSM.ground}
add_pst_state pst_ps_1 -pst pst -state {S1 S2 S3 S5}
add_pst_state pst_ps_2 -pst pst -state {S1 S2 S4 S5}
>>>>
```

When this variable is set to *true*, below is how the characterized UPF for the block "mid_inst" will look like. As can be seen, the derived PST has only one row. "*" is listed for SST.power, SST.ground and SSM.power, as all the states for these supplies are considered in the derived PST. SSM.ground has two states, S5 and S6. As only S5 is considered in the derived PST, "*" is not listed for SSM.ground.

```
>>>>
create_pst pst -supplies {SST.power SST.ground SSM.power SSM.ground}
add_pst_state pst_ps_1 -pst pst -state {* * * S5}
>>>>
```

---

upf_charz_create_pst_with_internal_state_names

Controls whether characterize should create a derived PST with internal state names for the block being characterized.

Data Types

Boolean

Description

This variable is applicable only when the design has at least one block with state propagation disabled.

When this variable is set to *false* (the default), characterize will create a derived group plus a derived PST for the block being characterized.

When this variable is set to *true*, characterize will create a derived PST with internal state names for the block being characterized. A derived group will not be created.

Example: Consider the following full-chip UPF -

```
>>>>
set_design_attributes -elements {.} -attribute
 enable_state_propagation_in_add_power_state false

create_supply_set SS1
```

```
create_power_domain PDTOP -supply {primary SS1}
add_power_state SS1 -state S1 {-supply_expr {power == {FULL_ON 0.9} &&
 ground == {FULL_ON 0.0}}}

create_supply_set SS2
create_power_domain PDMID -supply {primary SS2} -elements {mid_inst}
add_power_state SS2 -state S2 {-supply_expr {power == {FULL_ON 1.0} &&
 ground == {FULL_ON 0.0}}}
>>>>
```

When this variable is set to *false* (the default), below is how the characterized UPF for the block will look like. As can be seen, a derived group will be created for the block.

```
>>>>
create_supply_set SS1
create_supply_set SS2
add_power_state SS1 -state S1 {-supply_expr {power == `{FULL_ON, 0.9} &&
 ground == `{FULL_ON, 0.0}}}
add_power_state SS2 -state S2 {-supply_expr {power == `{FULL_ON, 1.0} &&
 ground == `{FULL_ON, 0.0}}}
create_power_state_group group
add_power_state -group group -state group_ps_1 {-logic_expr {SS1 == S1 &&
 SS2 == S2}}
>>>>
```

When this variable is set to *true*, below is how the characterized UPF for the block will look like. As can be seen, a derived PST with internal state names will be created for the block. In other words, the original supply set states will be brought into the block as port states.

```
>>>>
create_supply_set SS1
create_supply_set SS2
add_port_state SS1_power_port -state {SNPS_INT_S1_3 0.900000}
add_port_state SS1_ground_port -state {SNPS_INT_S1_4 0.000000}
add_port_state SS2_power_port -state {SNPS_INT_S2_7 1.000000}
add_port_state SS2_ground_port -state {SNPS_INT_S2_8 0.000000}
create_pst pst -supplies [list SS1_power_port SS1_ground_port
 SS2_power_port SS2_ground_port]
add_pst_state pst_ps_1 -pst pst -state {SNPS_INT_S1_3 SNPS_INT_S1_4
 SNPS_INT_S2_7 SNPS_INT_S2_8}
>>>>
```

Note: In the above UPF excerpt, SS1 is connected to the supply ports SS1_power_port and SS1_ground_port. Similarly for SS2.

This variable is not applicable when the entire design has state propagation enabled. In this case, characterize will anyway create a derived PST for the block. This derived PST will not have any internal state names.

Feedback

upf_charz_enable_domain_rescoping

Allows the *characterize* command to work on instances which contain power domains that correspond to a higher scope.

Data Types

Boolean

Default false

Group

none

Description

When this variable is set to *false* (the default), the *characterize* command will fail if used on a cell instance that contains a power domain belonging to a scope above that instance.

In order to characterize this sort of instance, this variable must be set to *true*. This will enable *characterize* to change the scope of the subdomain to the scope of the instance being characterized.

Domains which have their scope changed in this manner are not eligible for later domain merging by *propagate_constraints*.

Examples

Consider a simple design with the domain partitioning scheme:

```
create_power_domain TOP -include_scope
create_power_domain MID -elements {x/y}
```

Normally, characterize of the instance *x* is not permitted:

```
prompt> characterize x
Warning: Characterization of power supply data (UPF constraints,
 operating_voltages, etc.) to hierarchical cell 'x' have been skipped.
 (MV-116)
Reason: a descendent cell 'x/y' of 'x' belongs to a power domain 'MID'
 whose scope is above 'w'.
Error: Characterize failed. (CHR-002)
0
```

When the variable is set to *true*, the operation succeeds and the domain is re-scoped:

```
prompt> set upf_charz_enable_domain_rescoping true
true
prompt> characterize x
Information: The power domain 'MID' has a scope above 'x'. The scope of
 the domain will be changed in the subdesign.
```

```
  Characterizing cell 'x' on design 'mid'
1
```

See Also

- characterize

---

upf_charz_enable_supply_port_punching

Allows UPF-related supply port punching to occur within the *characterize* command on the blocks being characterized.

Data Types

Boolean

Default true

Group

mv

Description

The *upf_charz_enable_supply_port_punching* variable, when set to *true*, allows the *characterize* command to modify the block interfaces to automatically punch supply ports to bring required supply nets into the block if they are not already available.

See Also

- characterize

---

upf_charz_max_srsn_messages

Sets the maximum number of error and warning messages related to the *set_related_supply_net* command that can be printed when characterizing a block.

Data Types

int

Default 10

Group

upf

Feedback

Description

To reduce the verbosity of the *characterize* command, a limit is set on the number of messages related to the *set_related_supply_net* command that can be printed during characterization. You can change the verbosity of these messages by assigning an integer value to this variable.

See Also

- characterize
- set_related_supply_net

---

upf_check_bias_supply_connections

Determines whether bias connections on a supply net are checked for conflicting PG types.

Data Types

Boolean

Default true

Group

mv

Description

This variable determines whether bias connections on a supply net are checked for conflicting PG types.

By default, PG pins of the following types cannot be connected to the same supply net:

1. power and ground
2. nwell and pwell
3. nwell and ground
4. pwell and power

If the variable is set to *false*, only connections of type "a" are disallowed; bias supply connections (types "b," "c," and "d") are not checked.

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

prompt> *printvar upf_check_bias_supply_connections*

Feedback

upf_create_implicit_supply_sets

Allows creation of supply set handles for the power domains.

Data Types

Boolean

Default true

Description

This variable is set to *true* to enable the creation of supply set handles. When this variable is *true*, the tool creates the primary, default_isolation, and default_retention supply set handles when creating the power domains. Other user-defined supply set handles defined with the *-supply* option of the *create_power_domain* command are also created.

Note:

You must set this variable before creating the power domains. After creating the power domains, this variable is considered read-only. The tool issues the *CMD-013* error message if you change the value of the variable after creating the power domain.

See Also

- create_power_domain

upf_derive_ao_supply_on_exception_conns

Allow always-on derivation to rederive a new supply on a buffer with an exception connection

Data Types

Boolean

Default false

Group

upf

Description

When this variable is set to *true*, the tool will ignore explicit *connect_supply_net* statements which connect a supply to the backup pin of always_on buffers, if it can derive a better supply (or if no backup supply is necessary).

Feedback

When the variable is set to *false* (the default), the tool will treat explicit *connect_supply_net* statements connecting to the backup pin of always_on buffers as fixed supplies, and will never derive a better supply or convert the buffer to a non-always_on buffer.

See Also

- set_always_on_strategy
- connect_supply_net

---

upf_disable_b2b_iso_nor_optimization_strategies

Lists the isolation strategies for which back-to-back NOR optimization must be disabled.

Data Types

String

Default *

Group

upf

Description

This variable provides a list of isolation strategies for which back-to-back NOR optimization must be disabled.

Value of *upf_disable_b2b_iso_nor_optimization_strategies* is a space seperated string with each token in <power_domain>.<isolation_startegy_name> format. First name in the token is full name of power domain. Second name in the token is isolation strategy name. Wildcard characters other than * are not supported. Regular expressions are not supported.

```
When upf_disable_b2b_iso_nor_optimization_strategies is set to
"*"
   - Back-to-back NOR optimization is disabled for all isolation
 strategies
     in the design.
""
   - Back-to-back NOR optimization is enabled for all isolation
 strategies
     in the design.
"PD.ISO"
   - Back-to-back NOR optimization is disabled for isolation strategy ISO
 of
     power domain PD
"PD.ISO mid/PD.ISO"
   - Back-to-back NOR optimization is disabled for
```

```
    - isolation strategy ISO of power domain PD defined at TOP scope
    - isolation strategy ISO of power domain PD defined at mid scope
```

Variable is a overwrite variable, provided the new value is in the string format specified above. Variable value once set is valid for the entire Design Compiler session. This variable is applicable only to DC and has no impact on ICC shell.

All the isolation mapping commands like compile_ultra, compile_ultra -incr and insert_mv_cells will honor this variable setting.

If this variable is set post-compile, then it will be honored by the next mapping command for mapping newly inserted isolation cells. Existing isolation cells in the design will not be fixed as per this new setting.

This variable setting will not be written out in UPF' and also will not be written out in characterized block UPF.

Examples

In the following UPF

```
prompt> create_power_domain PDTOP
prompt> set_isolation ISO -domain PDTOP -applies_to outputs
prompt> set_scope mid_inst
prompt> create_power_domain PDMID -elements {mid_inst}
prompt> set_isolation ISO -domain PDMID -applies_to outputs
prompt> set_scope /

prompt> set upf_disable_b2b_iso_nor_optimization_strategies "PDTOP.ISO"
       // Back-to-Back NOR optimization is disabled for PDTOP.ISO and
       // will be implemented for mid_inst/PDMID.ISO

prompt> set upf_disable_b2b_iso_nor_optimization_strategies "PDB_ISO"
Error: Variable format is not acceptable: PDB_ISO is not in
 <domain_name>.<isolation_name> format. (UPF-821)
       // Previous setting will hold good
       // Back-to-Back NOR optimization is disabled for PDTOP.ISO and
       // will be implemented for mid_inst/PDMID.ISO

prompt> set upf_disable_b2b_iso_nor_optimization_strategies "PDTOP.ISO
 PDB.ISO"
       // Action commands will dump a warning saying PDB power domain is
 not defined.
       // Back-to-Back NOR optimization is disabled for PDTOP.ISO and
       // will be implemented for mid_inst/PDMID.ISO
```

See Also

- compile_ultra
- insert_mv_cells

Feedback

upf_drop_conflict_retention_constraint

Drop Retention constraint which is causing existing implemented retention registers dissociation

Data Types

Boolean

Default false

Group

upf

Description

When this variable is set to true, new Retention constraint causing existing implemented retention registers dissociation will be dropped.

Examples

The following example explain the usage model of the variable:

```
prompt> set upf_drop_conflict_retention_constraint true
prompt> set_retention RET -domain PDT \
    -elements { mid }
prompt> compile_ultra
prompt> set_retention RET -domain PDT \
    -exclude_elements { mid/reg } -update

RET update will be dropped because that will cause mid/reg dissociation
 from RET.
```

See Also

- set_retention
- map_retention_cell
- compile_ultra

upf_enable_legacy_block

Allows UPF 1.0-style design blocks to be integrated into a UPF 2.0-style design.

Data Types

Boolean

Default true

Feedback

Group

mv

Description

The *upf_enable_legacy_block* variable, when set to *true*, allows the *legacy_block* attribute to be applied to a UPF 1.0-style design instance through the *set_design_attribute* command. The *legacy_block* attribute on a UPF 1.0-style design instance allows a UPF 2.0-style domain-independent supply net outside the block to be connected to a domain-dependent net inside at the block boundary. The connection of a domain-independent supply net to a domain-dependent one is illegal otherwise.

---

upf_enable_mv_merge_clone

Controls the clone optimization of power management cells.

Data Types

Boolean

Default false

Group

none

Description

Setting the variable to *true* enables the clone optimization of power management cells.

See Also

- compile_ultra

---

upf_enable_relaxed_charz

Allows flexible partitioning of power domain and supports *-location parent* for isolation cells in a hierarchical flow.

Data Types

Boolean

Default true

Group

mv

Feedback

Description

The *upf_enable_relaxed_charz* variable, when set to *true*, allows the *characterize* command to handle *-location parent* for isolation cells and also allows greater flexibility in partitioning the design. The *characterize* command can partition a part of power domain.

See Also

- characterize

---

upf_extension

Disables writing UPF extension commands in the *save_upf* command.

Data Types

Boolean

Default true

Group

mv

Description

This variable controls whether the *save_upf* command writes UPF extension commands such as *set_related_supply_net* into UPF files. By default, UPF extension commands are written by *save_upf* in the following format:

```
if {[info exists upf_extension] && upf_extension} {
<upf_extension_command>
}
```

If *upf_extension* is set to *false*, these lines are not written by *save_upf*. The UPF extension command itself continues to be written by the *write_script* command, and continues to be readable by the *source* and *load_upf* commands.

All tools, including third party tools, that need to support the UPF extension commands must predefine the *upf_extension* Tcl variable as *true*.

See Also

- set_related_supply_net

---

upf_generate_pm_cell_html

Controls HTML reporting of unmapped power management cells when compile_ultra (-incr), insert_mv_cells and insert_dft commands are run.

Feedback

Data Types

Boolean

Default Variable is not supported in Design Compiler non-topographical mode, Design Compiler topographical mode, and DC Explorer

false in Design Compiler NXT non-topographical mode and Design Compiler NXT topographical mode

Group

mv

Description

When this variable is set to true, compile_ultra (-incr) and insert_dft commands generate HTML report for unmapped isolation, enable level shifter and retention cells. Whereas, insert_mv_cells command generates HTML report for unmapped isolation and enable level shifter cells. HTML report is produced only if there are unmapped Power Management cells in the design.

HTML report will list the reasons why each unmapped power management cell could not be mapped. Along with every reason, it will also list the library cells that failed to map to the cell due to that reason. HTML report directory called *'pm_cells_map_failure.<x>'* is dumped into the current working directory every time any of the aforementioned commands is issued and there are unmapped power management cells in the design. Here '<x>' is a number that starts with 0 and increments every time a new HTML report directory is created.

The default value is *false*, in which case no HTML reports are produced.

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

```
prompt> printvar upf_generate_pm_cell_html
```

---

upf_imvc_no_remap_iso

Specifies that the insert_mv_cells command should not remap non-GTECH cells.

Data Types

Boolean

Default false

Group

mv

Feedback

Description

This variable prevents non-GTECH cells from being remapped during the incremental behavior of the insert_mv_cells command. The tool can ignore an isolation cell instance if it's mapped to a valid isolation library cell. A valid isolation library cell:

- Complies with any restrictions related to the isolation strategy that references it. For example, restrictions set by the map_isolation_cell command or NOR-style isolation requirements

- Hass to match the design operating conditions of the referenced isolation cell.

The tool does not require the isolation cell match any other constraints such as target library subset or link library subset constraints. The tool does not perform conversions, such as isolation cell to enable level-shifter cell, if the isolation cell is mapped to a valid isolation library cell. This means that if there is a voltage violation, it is left unresolved.

To determine or change the current value of this variable, use get_app_var or set_app_var, respectively.

See Also

- insert_mv_cells
- map_isolation_cell

---

upf_infer_complex_retention_cells

Controls inference of complex retention cells in place of complex non-retention cells having retention strategies.

A cell is considered complex if it does not have a valid functionality.

Data Types

Boolean

Description

When this variable is set to *false* (the default), tool will not infer any complex retention cells in place of pre-instantiated complex non-retention cells. The pre-instantiated complex non-retention cells will be left as is in the final netlist.

When this variable is set to *true*, tool will infer complex retention cells in place of pre-instaniated complex non-retention cells having retention strategies. This style of inference will be supported both in *compile_ultra* and *compile_ultra -incremental*.

Feedback

Tool will try to infer a complex retention cell during *compile_ultra* based on pin name matching. Here are the steps that will be performed by the tool during *compile_ultra*:

1. Pin names of the non-retention cell will be matched with the pin names of the retention library cells specified in the *map_retention_cell* command.
2. The first retention library cell whose pin names exactly match (barring the retention save/restore pins) will be picked up and swapped in place of the non-retention cell.
3. If a matching retention library cell is not found, then a warning message will be printed saying that the non-retention cell cannot be converted to retention.

With the pin name matching approach, the tool will pick up the first matching retention library cell from the *map_retention_cell* command. If there are multiple matches available in *map_retention_cell*, and if the user wants a particular retention library cell to be picked up among the matches, then users will be able to achieve this by setting a new attribute called *retention_equivalent* on the non-retention library cell. The value of this attribute will be the name of the preferred (equivalent) retention library cell.

Usage -

```
        set_attribute {library/non_retention_library_cell_name}
retention_equivalent {retention_library_cell_name}
```

If the retention library cell (specified as the attribute value) is invalid, then the tool will fall back to the pin name matching approach. A retention library cell will be considered invalid in the following scenarios:

1. The retention library cell is not present in any of the target libraries.
2. The retention library cell has the dont_use attribute on it.
3. The retention library cell does not have the *retention_cell* attribute on it.
4. The retention library cell is not present as one of the library cells in the *map_retention_cell* command.
5. The non-retention library cell has at least one pin that is not present on the retention library cell.

So, if an invalid retention library cell is found as the attribute value, then the tool will not honor the *retention_equivalent* attribute, and it will fall back to the pin name matching approach.

Point # 5 means that all the pins present on the non-retention library cell must also be present on the retention library cell. But it is okay for the retention library cell to have additional pins (other than the retention save/restore pins). Examples for additional pins are scan input and scan enable pins. If a retention library cell having additional pins is

inferred by the tool, then *compile_ultra* will connect the additional pins to constant zero. If a particular pin is not supposed to be connected to constant zero, then the user is expected to fix the connection after *compile_ultra* (before taking the netlist further down the flow). Whenever a retention library cell having additional pins is inferred, a message will be printed to let the users know about such an inference. This should help the users to watch out for the connections made by the tool to the additional pins. Inferring retention cells with additional pins will be helpful for users who want to replace a non-scan non-retention cell (instantiated in the RTL) with a scan retention cell. insert_dft can later do the required scan stitching with the scan input/scan enable pins.

Handling mismatching pin names: Users may have library cells where certain pin names of the non-retention library cell do not match the pin names of the retention library cell. In such cases, users can use the same *retention_equivalent* attribute to specify the pin mappings between the mismatching pin names.

Usage -

```
        set_attribute {library/non_retention_library_cell_name}
retention_equivalent {retention_library_cell_name {pin1 pin2} {pin3
pin4}}
```

In this usage, the attribute value has the retention library cell name listed first and then the pin mappings between the non-retention library cell and the retention library cell. The above usage means that:

1. *pin1* on the non-retention library cell corresponds to (or is equivalent to) *pin2* on the retention library cell.
2. *pin3* on the non-retention library cell corresponds to *pin4* on the retention library cell.

This information will help the implementation tools to transfer the net connections appropriately. If the pin mappings are invalid, then the tool will fall back to the pin name matching approach. Pin mappings will be considered invalid in the following scenarios:

1. The specified pin name does not exist on the non-retention library cell/retention library cell.
2. Other than the specified pin mappings, the non-retention library cell has at least one pin that is not present on the retention library cell.

---

upf_insert_clamp_in_zpr_hierarchy

Controls the hierarchical location of zero-pin retention clamp cells.

Data Types

Boolean

Feedback

Description

When this variable is set to *true*, zero-pin retention clamp cells will always be inserted into the same hierarchy as the retention cells they are clamping.

When this variable is set to *false* (the default), zero-pin retention clamp cells may be inserted in higher-level hierarchies, so that they may be shared by retention cells in multiple lower hierarchies.

See Also

- set_retention

---

upf_iso_filter_elements_with_applies_to

Controls the filtering of design elements specified in the *-elements* option when using the *-applies_to* option of the *set_isolation* command.

Data Types

string

Default ENABLE

Group

mv

Description

This variable controls the filtering of design elements specified in the *-elements* list based on the direction specified in *-applies_to* option of the *set_isolation* command.

The default is *ENABLE*, which allows the usage of the *-elements* and *-applies_to* together in the *set_isolation* command. The elements are filtered based on the direction specified by the *-applies_to* option.

Setting the variable to *DISABLE* instructs the tool to ignore the *-applies_to* option and apply the isolation strategy to all the specified elements.

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

```
prompt> printvar upf_iso_filter_elements_with_applies_to
```

See Also

- set_isolation

Feedback

upf_iso_insert_iso_ctrl_inverters_in_other_hier

Specifies whether to insert isolation control inverters in a different logical hierarchy from their respective isolation cells.

Data Types

Boolean

Default true

Group

mv

Description

Depending on isolation strategies' sense values and lib_cell availability, it may be necessary for the tool to insert inverters on the isolation control pins of some isolation cells. When this variable is set to *true* (the default), the tool may place control inverters in a different logical hierarchy from their respective isolation cells. It will do this if an always-on inverter is required but no such inverter is available, or if an inverter is required in an outer hierarchy and can be reused in an inner one.

If the option *upf_map_illegal_control_inverters* is set to *false*, this option will be automatically disabled.

See Also

- set_isolation
- upf_map_illegal_control_inverters

upf_iso_map_exclude_zpr_clamp_lib_cells

Controls library cells used to map regular isolation cell when mapping constraint is specified for zero-pin retention clamp cells.

Data Types

string

Default false

Group

mv

Feedback

Description

This variable controls the library cells used to map regular isolation cells when a mapping constraint has been specified to map zero-pin retention clamp cells with *map_retention_clamp_cell* command.

There are three legal values this variable can be set to - "true", "false", "nor". The default value is *false*, which allows library cells specified in map_retention_clamp_cell command to be used to map regular isolation cells. When this variable is set to true, tool will not pick library cells used in map_retention_clamp_cell to map regular isolation cells. When this variable is set to nor, tool will only exclude NOR isolation library cells specified in map_retention_clamp_cell command when mapping regular isolation cells. However, if a library cell is specified with both *map_retention_clamp_cell* and *map_isolation_cell* command, *map_isolation_cell* gets higher precedence, and the library cell can still be used to map regular isolation cells of the isolation strategy. Global control does not play any role in this scenario.

If the design has regular isolation cells mapped to a library cell mentioned in *fBmap_retention_clamp_cell* command, the next mapping command will remap this cell to a library cell not specified as part of *map_retention_clamp_cell* command.

```
prompt> printvar upf_iso_map_exclude_zpr_clamp_lib_cells
```

See Also

- map_retention_clamp_cell

---

upf_iso_skip_single_rail_for_non_primary_iso_supply

Controls tool behavior when necessary isolation lib_cells are not available.

Data Types

Boolean

Default false

Description

When mapping isolation cells, the tool will select lib_cells based on the local power intent. This variable controls the behavior of the tool when the local intent requires dual-rail isolation cells, but only single-rail isolation cells are available in the library.

If this variable is set to true, isolation cells will be left unmapped (as GTECH cells), if dual-rail lib_cells are required but not available. If it is left as false (the default), isolation cells will be mapped as single-rail cells.

Feedback

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

```
prompt> printvar upf_iso_skip_single_rail_for_non_primary_iso_supply
```

See Also

- set_isolation
- map_isolation_cell

---

upf_isolation_enable_relax_self_dependency_check

Controls the mode (warn vs error) of self dependency check of an isolation control.

Data Types

Boolean

Default true

Group

none

Description

Setting the variable to *false* enforces the self dependency check to be error.

UPF: set_isolation ISO1 -domain PD_TOP -elements {CTRL} -isolation_signal {CTRL}

CTRL has a self dependency because CTRL enable signal is also CTRL. Default tool will flag a warning. When the variable is set to *false* tool will flag an error for this case.

See Also

- set_isolation

---

upf_isols_allow_instances_in_elements

Controls the specification of instances in the *-elements* option of the *set_isolation* command.

Data Types

Boolean

Default true

Feedback

Group

mv

Description

This variable controls the specification of instances as the target elements for the *set_isolation* command. By default, you can specify any hierarchical cell, macro, or black box as the target element for isolation.

Set this variable to *false* if you do not want the *set_isolation* command to honor the specification of instances in its *-elements* list.

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

```
prompt> printvar upf_isols_allow_instances_in_elements
```

See Also

- set_isolation

---

upf_levshi_on_constraint_only

Inserts level shifters only at the domain boundaries with a level shifter strategy.

Data Types

Boolean

Default false

Group

mv

Description

This variable controls whether level shifter cells can be inserted at power domain boundaries without a strategy. By default, level shifters can be inserted at domain boundaries without a strategy. This gives more flexibility to the tool and has better QoR.

If you want to restrict the tool to insert level shifters only at the boundaries where the *set_level_shifter* command is specified, set the variable to *true*. The tool will not insert level shifters at all of the boundaries with *set_level_shifter*. The tool can insert level shifters only at the boundaries with the *set_level_shifter* constraint.

See Also

- set_level_shifter

Feedback

upf_ls_strategy_in_inst_name

Controls if level shifter strategy name should be included in level shifter and enable level shifter cell names.

Data Types

Boolean

Default false

Description

This variable controls whether level shifter strategy name and power domain should be included in level shifter and enable level shifter cell names. on output pins. The default value of this variable is *false*, level shifter and enable level shifter cells will not have level shifter strategy and power domain name included in cell name.

upf_map_illegal_control_inverters

Controls behavior of isolation insertion when always-on control inverters are not available

Data Types

Boolean

Default true

Group

mv

Description

When implementing isolation strategies, depending on the available libraries, it may be necessary to add always-on inverters on the isolation control signal. If always-on inverters are not available in the target library, there are two possible behaviors. If this variable is set to *true*, then regular inverters will be used instead, leading to *MV-076* errors. If this variable is set to false, then in absence of always-on inverters, GTECH isolation cells will be used and user will see unmapped isolation cells.

See Also

- set_isolation
- set_isolation_control

Feedback

upf_name_map

Specifies the name map files to be used during reapplication of golden UPF to designs.

Data Types

list

Default ""

Group

mv

Description

This variable specifies the name of the mapping file to be used for each design during reapplication of golden UPF to the designs. The variable can specify a single design name and its corresponding map file, or it can list multiple pairs of design names and corresponding map file names.

For example, to set a single design name and map name:

```
prompt> set upf_name_map [list {my_design my_design.map}]
```

To set multiple designs and corresponding map names:

```
prompt> set upf_name_map [list {top top.map} \\
        {mid mid.map}]
```

In a golden UPF session, the tool can change object names as a result of ungrouping and expansion of wildcard names to full names. When the golden UPF and supplemental UPF are applied to a design in a later tool session, the object names used in the golden UPF file must be properly mapped to the new names used in the design netlist. Synopsys Galaxy flow tools all use the same renaming conventions to keep track of changed names.

However, the default renaming conventions cannot handle some complex renaming situations. In those cases, the *write -format verilog* command, in addition to writing out the Verilog netlist, also writes an explicit name mapping file containing the renaming rules for the specific design. It also inserts a link to the name mapping file into the Verilog netlist, in the form of a pragma statement.

In a later tool session, when you read in this Verilog netlist, the tool automatically finds the applicable name map file and properly maps the object names, and it is not necessary to set the *upf_name_map* variable. However, if the name map file has been moved to a new location, or if the Verilog file no longer contains the pragma statement, you can set this variable to specify the location of the name mapping file. The variable setting overrides any conflicting location specified by a pragma in the Verilog netlist file.

Feedback

The name mapping file is a Tcl script containing Tcl built-in commands and two Synopsys commands: *set_query_rules*, which defines renaming rules for rule-based query, and *define_name_maps*, which defines the name mapping for specific objects. The file is provided information only; you should not edit or modify it.

The following is an example of a name mapping file:

```
# Query Rules
set_query_rules hierarchical_separator {/ _} bus_notation {[] __}

# Tcl built-in: local variables
set hier_1  A/B/C/D
set hier_2  A_B/B2/this/is/a/long/path

# Explicit name maps
define_name_maps \\
     -application golden_upf \\
     -design_name design \\
     -columns {class pattern options names} \\
     [list cell $hier_1/A_reg [list leaf]        [list $hier_2/ger_A]
     ] \\
     [list cell A/B/X*_reg    [list nocase leaf] [list A_B/zar_reg
 A_B/Y_reg3]]

# cleanup
unset hier_1
unset hier_2
```

To determine the current value of this variable, use *printvar upf_name_map* command. For a list of all mv variables and their current values, use the *print_variable_group mv* command.

See Also

- load_upf
- save_upf
- enable_golden_upf
- define_name_maps
- set_query_rules
- write

upf_nor_iso_macro_allow_enable_supply_check

Controls if on-ness checks are to be done for all enable pins of nor(or nand) isolated macro output pins with multi-input isolation enable condition.

Feedback

Data Types

Boolean

Default true

Description

This variable controls how the tool performs on-ness checks between enable pins of a nor(or nand) isolated macro output pin. By default the value of this variable is *true*. A warning message (MV-570) is issued if any enable pin is less always on or unrelated to one(or more) load pin(s).

User has a way to avoid this warning by setting this variable to *false*. In this case, tool only requires at lease one enable pin be equal or more always on than all load pins.

---

upf_pg_writer_output_all_supply_ports

Controls the feature to write out supply ports which don't drive leaf cells to the PG netlist.

Data Types

Boolean

Description

When this variable is set to *true*, "write_file -format verilog -pg" will write out all UPF supply ports and nets defined in the design, even those which do not connect to the PG pins of any cell instances in the design.

When this variable is set to *false* (the default), "write_file -format verilog -pg" will write out UPF supply ports and nets which eventually drive the PG pins of leaf cells and macros.

---

upf_pm_data_net_force_scalar

Remove signal nets connected to an isolation cell data pin from their original bus to handle them separately.

Data Types

Boolean

Default false

Group

mv

Feedback

Description

When set to true the tool will remove, from any signal net connected to a tool inserted isolation cell, any attribute that identifies it as part of a data bus.

This will stop *change_names* from renaming affected nets based on bus naming related rules, also the net will not be written as part of a bus when writing out the resulting verilog file.

See Also

- change_names
- write

---

upf_power_model_library

This variable specifies a list of power model UPF files tool should read.

Data Types

String

Default ""

Description

This variable specifies a list of power model UPF files in the directory specified by the variable upf_power_model_search_path that tool should read at the beginning of the load_upf command. Only the define_power_model command in these UPF files will be processed.

This variable should be set before the first load_upf command is executed.

Example:

```
prompt> set upf_power_model_library "model_1.upf model_2.upf"
```

See Also

- define_power_model
- apply_power_model
- report_power_model
- upf_power_model_search_path

Feedback

upf_power_model_search_path

This variable specifies the path to search for power model UPF files.

Data Types

String

Default ""

Description

This variable specifies a list of directory paths that store power model UPF files. Tool should search for power model UPF files specified by variable upf_power_model_library in this list of directory paths.

Power models will be read in the same order as directory paths and library files appear in these variables. Duplicated power models with the same names will be skipped.

This variable should be set before the first load_upf command is executed.

Example:

```
prompt> set upf_power_model_search_path
 "/usr/lib/power_model_lib ./power_model_lib"
```

See Also

- define_power_model
- apply_power_model
- report_power_model
- upf_power_model_library

upf_power_switch_track_ctrl_ack_pins

Store the original input provided to the -control_port / -ack_port arguments to create_power_switch.

Data Types

Boolean

Default false

Group

none

Feedback

Description

The *create_power_switch* UPF command supports the *-control_port* and *-ack_port* arguments, to specify logic nets which correspond to the control and acknowledge ports of the power switch.

If this argument contains the name of a logic pin, instead of a net, the tool will resolve the argument to the net connected to the given pin.

When this variable is *true*, the tool will keep track of any pins specified as *-control_port* and *-ack_port* arguments, and when a UPF' file is created with the *save_upf* command, the *create_power_switch* statement will contain the original pin name. When this variable is *false* (the default), the UPF' output will contain the name of the resolved net segment.

See Also

- create_power_switch
- save_upf

---

upf_preserve_logic_in_boolean_expr

preserve logic signal referenced in the *-logic_expr* option of the *set_port_attributes command*

Data Types

Boolean

Default false

Group

mv

Description

Normally, the tool makes no effort in preserving any logic signals referenced in the *-logic_expr* option of the *set_port_attributes* command, dropping the expression if it becomes invalid due to the tool optimizing away some of the logic elements referenced in it.

Setting this variable to true restricts the tool from performing optimization or removing any logic elements referenced in the *-logic_expr* option of the *add_power_state* command, so that the statement remains valid during the flow. This might impact QoR.

See Also

- add_power_state

Feedback

upf_print_states_with_voltages

Controls whether state names will be printed in MV-231 message.

Data Types

Boolean

Default false

Description

This variable controls whether state names will be printed along with voltage values in MV-231 message. The default value of this variable is *false*.

When this variable is set to *false* (default value), only voltage values are printed in MV-231 message.

Example:

```
prompt> check_mv_design
Warning: Pin 'ABC'(SN1[0.86,1.08]) cannot drive 'XYZ'(SN2[0.86,1.08]) due
 to PST voltage range differences (effective strategy is [rule = both,
 threshold = 0.00]). (MV-231)
```

When this variable is set to *true*, state names are also printed along with the voltage values in MV-231.

Example:

```
prompt> check_mv_design
Warning: Pin 'ABC'(SN1[state_0_8(0.86),state_1(1.08)]) cannot drive
 'XYZ'(SN2[state_0_8(0.86),state_1(1.08)]) due to PST voltage range
 differences (effective strategy is [rule = both, threshold = 0.00]).
 (MV-231)
```

upf_proceed_on_bias_rail_order_error

Allows the *compile* and *insert_mv_cells* commands to proceed when bias rail order errors are found

Data Types

Boolean

Default false

Group

mv

Feedback

Description

When set to *true* tool will proceed when bias rail order errors are reported (e.g UPF-659), instead of stopping until the issues are fixed.

---

upf_relax_target_library_subset_for_pm_cells

Set this variable to relax set_target_library_subset constraint.

Data Types

Boolean

Default false

Group

upf

Description

To map the power management cells, tool abides by both UPF constraints and target_library_subset constraint. If there is no overlap between UPF and target_library_subset libcells, tool leaves the power management cells unmapped or level shifter violations unaddressed. Thus, setting the variable will help relaxing the set_target_library_subset constraint and use only UPF constraint during mapping. This helps avoiding unmapped power management cells. Currently, the variable takes effect only for isolation, level shifter and enable level shifter cells.

Examples

```
Consider matching isolation library cells present in the target libraries
 are, ISO_1, CKISO

UPF:
prompt> set upf_relax_target_library_subset_for_pm_cells true
prompt> set_target_library_subset -clock_path -dont_use {CKISO} -top
prompt> create_power_domain PD1
prompt> set_isolation iso_mid1 -domain PD1
prompt> map_isolation_cell -lib_cells {CKISO}
prompt> compile_ultra

With the variable set to false, tool will not consider libcell
CKISO for isolation mapping as it is marked as dont_use via
set_target_library_subset command. However, with the variable
set to true, set_target_library_subset constraint will be relaxed
and tool will consider CKISO libcell for isolation mapping.
```

Feedback

See Also

- set_target_library_subset

---

upf_report_isolation_matching

Specifies whether to report occurrences of an isolation strategy being applied to an existing isolation cell.

Data Types

Boolean

Default false

Group

mv

Description

The *upf_report_isolation_matching* variable, when set to *true*, causes the tool to report a UPF-073 information message whenever an isolation strategy is applied to an existing isolation cell.

If an isolation strategy is specified for a port that already has an isolation cell, the tool verifies that the existing isolation cell satisfies all of the parameters of the isolation strategy, except the power and ground nets. If these parameters are satisfied, the tool uses the existing cell and does not create another isolation cell for the port.

Set this variable to *true* if you want occurrences of this condition to be reported as UPF-073 information messages.

See Also

- set_isolation

---

upf_retention_analyze_pin_precedence_of_lib_cells

Analyzes pin precedence of library cells and accordingly enables or disables sequential output inversion.

Data Types

Boolean

Default false

Feedback

Description

When this variable is set to *false* (the default), tool does not analyze pin precedence of library cells. In this mode, sequential output inversion is enabled on cells having retention strategies.

When this variable is set to *true*, tool analyzes pin precedence of library cells. If both set dominant/reset dominant and restore dominant retention library cells are found, then the tool disables sequential output inversion on cells having retention strategies. This is to make sure that a set/reset dominant library cell is not converted to a restore dominant library cell (or vice-versa) through sequential output inversion, as such a transformation can result in a verification failure.

---

upf_skip_ao_check_for_els_input

Specifies whether enable level shifters inserted by the tool are allowed to use a power supply that is more always-on than the driver supply.

Data Types

Boolean

Default true

Group

none

Description

When this variable is set to *true* (the default), enable level shifters inserted by the tool can use a power supply that is either equally always-on or more always-on than the supply for the driver of the inputs of the enable level shifter.

When this variable is set to *false*, enable level shifters inserted by the tool must use a power supply that is equally always-on compared to the supply for the driver of the inputs of the enable level shifter.

An enable level shifter is a power management cell that performs both isolation and level shifting between two power domains.

See Also

- compile_ultra
- insert_mv_cells

Feedback

upf_skip_retention_clamp_insertion

Controls the insertion of zero-pin retention clamp cells during compile flow.

Data Types

Boolean

Description

When this variable is set to *true* (the default), then zero-pin retention clamp cells will be inserted during compile, for any zero-pin retention cells which have been mapped in the design.

When this variable is set to *false*, clamp cell insertion is skipped.

This variable has no effect if *set_retention* strategies with zero-pin retention library cells are not used in the design.

See Also

- set_retention

upf_skip_retention_on_dft_cells

Skips Retention Register mapping of registers newly introduced by insert_dft

Data Types

Boolean

Default false

Group

upf

Description

When this variable is set to true, all the registers newly introduced by insert_dft will not undergo Retention mapping. They will be mapped to non-retention registers.

However, independent of the value set for this variable, scan replacement of existing Retention Registers that happens as part of insert_dft will continue to honor Retention constraints.

One element level *set_retention -no_retention* strategy per power domain will be generated for the newly introduced DFT registers on which retention mapping is skipped.

Feedback

A unique name will be generated as *snps_no_retention_<domain_name>* for each retention strategy for a given power domain. If a -no_retention strategy exists with name *snps_no_retention_<domain_name>* during insert_dft flow, the same will be updated to include newly inserted DFT cells.

compile_ultra and compile_ultra -incr will also honor this variable, if set to *true*.

Examples

The following example generates retention strategy for newly introduced DFT cells {SCAN1 SCAN2} in power domain PDT:

```
prompt> set upf_skip_retention_on_dft_cells true
prompt> set_retention snps_no_retention_PDT -domain PDT \
    -elements { SCAN1 SCAN2 } -no_retention
```

See Also

- set_retention
- compile_ultra
- insert_dft

---

upf_smart_derive_iso_strategy_on_new_control_ports

Controls the smart derivation of no_isolation strategy on newly punched ports on retention control path.

Data Types

Boolean

Default false

Group

none

Description

Setting the variable to *true* enables the smart derivation of no_isolation strategy.

See Also

- set_isolation
- set_retention

Feedback

upf_suppress_empty_strategy_messages

Suppresses warning messages caused by empty strategies during insertion.

Data Types

Boolean

Default false

Group

none

Description

During level shifter and isolation insertion, the tool checks for empty strategies (strategies with an empty element set) and display a warning for each one, setting this variable to *true* will stop said messages from beign displayed.

See Also

- UPF-912

upf_suppress_etm_model_checking

Disables model checking when referring to the UPF of a macro cell.

Data Types

Boolean

Default false

Group

mv

Description

This variable controls model checking on the UPF for a macro cell. By default, when using the *create_supply_net* and *create_supply_port* commands on a power domain defined at the scope of a macro cell, the tool checks consistency against the cell's power and ground pin definitions. By setting this variable to *true*, these checks are skipped.

See Also

- create_supply_net
- create_supply_port

Feedback

upf_suppress_message_in_black_box

Suppresses warning messages caused missing objects when UPF commands are loaded at the black box scope.

Data Types

Boolean

Default true

Group

none

Description

When loading UPF at scope of a black box (Verilog stub), netlist objects inside the block are not accessible. If the UPF for the black box contains commands that refer to netlist objects inside the block, those objects are ignored.

When this variable is set to *true* (the default), no warning message is displayed for missing objects. For example,

```
prompt> connect_supply_net sn -port sub/sp
prompt>
```

When this variable is set to *false*, warning messages are displayed for missing objects. For example,

```
prompt> set upf_suppress_message_in_black_box false
false
prompt> connect_supply_net sn -port sub/sp
Warning: Can't find supply ports or power pins matching 'sub/sp' in
 design 'mid'. (UID-95)
prompt>
```

See Also

- UPF-623
- UPF-626
- UPF-627

upf_suppress_message_in_etm

Suppresses warning messages when the current scope is an extracted timing model (ETM) and you use disallowed UPF commands.

Feedback

Data Types

Boolean

Default true

Group

none

Description

When loading UPF at the scope of an extracted timing model (ETM) instance, only the *create_power_domain*, *add_port_state* and *set_scope* UPF commands are allowed. Any other UPF commands are ignored.

When this variable is set to *true* (the default), no warning message is issued for ignored commands. For example,

```
prompt> create_supply_net sn
sn
prompt>
```

When this variable is set to *false*, the tool issues a warning message when disallowed commands are ignored. For example,

```
prompt> set_app_var upf_suppress_message_in_etm false
false
prompt> create_supply_net sn
Warning: UPF Command 'create_supply_net' is skipped under ETM scope
 'I_SUB'. (UPF-628)
prompt>
```

See Also

- UPF-628

---

upf_track_bias_supply_net_resolution

Controls whether the UPF commands associated to supply set bias function resolution are added to the output of *save_upf*

Data Types

Boolean

Default false

Feedback

Description

For UPF supply sets created under scopes marked the *enable_bias DERIVED* design attribute, tool will automatically resolve the *nwell* and *pwell* bias functions of the supply set to its *power* and *ground* functions respectively, if necessary to continue with the flow.

The *upf_track_bias_supply_net_resolution* variable can then be set to *true* to have the output of the *save_upf* command include the UPF commands generated by the tool for the bias function resolution process described above.

---

upf_use_driver_receiver_for_io_voltages

Controls if related supply of top level ports is derived from related supply of connected pad pins.

Data Types

Boolean

Default true

Description

This variable controls how the tool derives supply for top-level ports. By default the value of this variable is *true*. If a top-level port has one or more connected cell pins with is_pad attribute true, related supply will be derived from pad pins. This supply will be used even if user has set related supply onsuch ports with set_port_attributes or set_related_supply_net command in their UPF.

User has the option to not use pad supply for top-level ports by setting this variable to *false*.

---

upf_warn_on_logic_to_supply_object_upgrade

Controls printing warning on logic ports that are promoted to supply ports.

Data Types

Boolean

Description

If this variable is set to *true*, a warning is printed that the logic port is promoted to a supply port.

---

upf_write_highest_upf_version

Control writing out of UPF version when save_upf command is used.

Feedback

Data Types

Boolean

Default false

Group

mv

Description

This variable controls how UPF version is written out when save_upf command is used.

When the variable is set to *true*, during *save_upf* the tool will write out only the highest UPF version.

The default is *false*. When the variable is set to *false*, during *save_upf* the tool will write out all the UPF version commands.

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

```
prompt> printvar upf_write_highest_upf_version
```

Examples

The following example illustrates that only the highest *upf_version* command is written in the UPF saved.

```
prompt> set upf_write_highest_upf_version true
prompt> upf_version 2.1
IEEE-1801
prompt> upf_version 2.0
IEEE-1801
 ...
prompt> save_upf out.upf
1
```

The UPF file "out.upf" written by the *save_upf* command contains the command "upf_version 2.1".

See Also

- upf_version

---

upf_write_only_rtlpg_to_pg_netlist

Controls the feature to write out only the RTLPG connections to the PG Verilog netlist in the presence of UPF.

Feedback

Data Types

Boolean

Description

When this variable is set to *true*, "write -format verilog -pg" will write out only the RTLPG connections to the PG Verilog netlist in the presence of UPF.

When this variable is set to *false* (the default), "write -format verilog -pg" will write out the full PG Verilog netlist in the presence of UPF.

This variable needs to be set before reading in the RTL.

Below is how the flow will look like in the presence of this variable.

```
>>>>
set dc_allow_rtl_pg true
set upf_write_only_rtlpg_to_pg_netlist true

read_verilog

load_upf

convert_pg
write -format verilog -pg
>>>>
```

See Also

- dc_allow_rtl_pg

---

use_port_name_for_oscs

Specifies that when off-sheet connectors for nets also have ports on them, they are given the name of the port.

Data Types

Boolean

Default false

Description

This variable specifies that when off-sheet connectors for nets also have ports on them, they are given the name of the port.

By default, the connectors are given the name of the net.

Feedback

v

vao_feedthrough_module_name_prefix

Variable to specify the prefix for the name of new hierarchies created during voltage area aware always-on synthesis.

Data Types

string

Default ""

Group

mv

Description

In the voltage area aware always-on synthesis triggered by the *psynopt* or the *place_opt* command, new hierarchies are created to hold the newly inserted always-on buffers, to fix DRC violations. Using this variable you can specify prefix for the new hierarchy created during voltage area aware always-on synthesis.

Multicorner-Multimode Support

This variable has no dependency on scenario-specific information.

Examples

For the new hierarchies to start with the prefix SNPS_VAO, set the variable as follows:

set_app_var vao_feedthrough_module_name_prefix {SNPS_VAO}

verbose_messages

Causes more explicit system messages to be displayed during the current session.

Data Types

Boolean

Default true

Description

This variable causes more explicit system messages to be displayed during the current session.

Feedback

verilogout_equation

Writes Verilog "assign" statements (Boolean equations) for combinational gates, rather than gate instantiations.

Data Types

Boolean

Default false

Description

This variable writes Verilog "assign" statements (Boolean equations) for combinational gates, rather than gate instantiations.

By default, gate instantiations are written.

verilogout_higher_designs_first

Writes Verilog "modules" so that the higher-level designs come before lower-level designs, as defined by the design hierarchy.

Data Types

Boolean

Default false

Description

This variable writes Verilog "modules" so that higher-level designs come before lower-level designs, as defined by the design hierarchy. The default is to write lower-level designs first.

verilogout_ignore_case

Instructs the compiler not to consider the case when comparing identifiers to Verilog reserved words.

Data Types

Boolean

Default false

Group

hdl_variables

Feedback

Description

This variable instructs the compiler not to consider the case when comparing identifiers to Verilog reserved words.

When an identifier is equal to a reserved word, the identifier is escaped by putting a backslash (`\\`) in front of it. When this variable is set to *false*, case is considered. When set to *true*, case is ignored in the comparison.

Therefore, if *verilogout_ignore_case* is set to *true*, the default allows an identifier BUF to pass unchanged, BUF becoming \\BUF.

---

verilogout_include_files

Specifies to the *write -f verilog* command to write an include statement that will have the name of the value you set for this variable.

Data Types

list

Default ""

Group

hdl_variables

Description

This variable specifies to the *write -f verilog* command to write an include statement that will have the name of the value you set for this variable. For example, when you specify *verilogout_include_files*={"my_header.v"}, you see an include "my_header.v" in your Verilog output.

See Also

- write

---

verilogout_indirect_inout_connection

Uses nets snps_logic_zero and snps_logic_one to indirectly connect inout ports to Logic 0 or 1.

Data Types

Boolean

Default false

Feedback

Description

By default the Verilog Writer will write any port connected to Logic 0 or 1 as connected to "1'b0" or "1'b1". The Verilog Standard is more strict, however, and it asks that inout ports use an intermediate net for this kind of connections. When this variable is set to true the Verilog Writer will use snps_logic_zero and snps_logic_one nets, which are in turn assigned to their corresponding logic.

---

verilogout_inout_is_in

Instructs the tool to treat inout ports as inputs, changing from the default of output, in the Verilog output netlist. This can have an effect on simulation results if tristates are disabled by setting the *verilogout_no_tri* variable.

Data Types

Boolean

Default false

Description

Instructs the tool to treat inout ports as inputs in the Verilog output netlist.

---

verilogout_no_tri

Declares three-state nets as Verilog wire instead of tri. This variable is useful to eliminate assign primitives and tran gates in the Verilog output. If there are real inout ports in the design, you need tristates to be able to read and write to them without any direction or cycle issues in simulation; that is, tran gates allow interfacing two drivers. For synthesis, an assign statement is not directional. It is a simple wire used as an electrical connection, but for simulation it is directional and could result in simulation issues if you deactivate tristates.

Data Types

Boolean

Default false

Description

Declares three-state nets as Verilog wire instead of tri. Use this variable to eliminate assign primitives and tran gates in the Verilog output.

Feedback

verilogout_show_unconnected_pins

Instructs the Verilog writer to write out all of the unconnected instance pins, when connecting module ports by name. For example, modb b1 (.A(in),.Q(out),.Qn()).

Data Types

Boolean

Default false

Description

This variable instructs the Verilog writer to write out all of the unconnected instance pins, when connecting module ports by name. For example, modb b1 (.A(in),.Q(out),.Qn()).

By default, the Verilog writer does not write out any unconnected pins. For example, modb b1 (.A(in),.Q(out)).

verilogout_single_bit

Instructs the compiler not to output vectored ports in the Verilog output. All vectors are written as single bits.

Data Types

Boolean

Default false

Description

This variable instructs the compiler not to output vectored ports in the Verilog output. All vectors are written as single bits.

verilogout_unconnected_prefix

Instructs the Verilog writer to use the name SYNOPSYS_UNCONNECTED_ to create unconnected wire names. The general form of the name is SYNOPSYS_UNCONNECTED_%d.

Data Types

string

Default SYNOPSYS_UNCONNECTED_

Feedback

Description

This variable instructs the Verilog writer to use the name (SYNOPSYS_UNCONNECTED_) to create unconnected wire names. The general form of the name is SYNOPSYS_UNCONNECTED_%d.

The purpose of this variable is to avoid conflict with the name already used in the design.

---

vhdlout_bit_type

Sets the basic bit type in a design written to VHDL.

Data Types

string

Default std_logic

Description

The *vhdlout_bit_type* variable sets the basic bit type in a design written to VHDL. This is useful when you base your design methodology on a logic value system other than std_logic.

See Also

- vhdlout_bit_vector_type
- vhdlout_one_name
- vhdlout_three_state_name
- vhdlout_zero_name

---

vhdlout_bit_vector_type

Sets the basic bit vector type in a design written to VHDL.

Data Types

string

Default std_logic_vector

Description

The *vhdlout_bit_vector_type* variable sets the basic bit vector type in a design written to VHDL. This is useful when your design methodology is based on a logic value system other than std_logic.

Feedback

See Also

- vhdlout_bit_type
- vhdlout_one_name
- vhdlout_three_state_name
- vhdlout_zero_name

vhdlout_dont_create_dummy_nets

Instructs the VHDL writer not to create dummy nets to connect unused pins or ports in your design.

Data Types

Boolean

Default false

Description

The *vhdlout_dont_create_dummy_nets* variable instructs the VHDL writer that it is not to create dummy nets to connect unused pins or ports in your design.

If you want the VHDL writer to create dummy nets to connect unused pins or ports in your design, set the value of this variable as *false* (the default value).

vhdlout_equations

Defines how the tool is to write combinational logic and sequential logic.

Data Types

Boolean

Default false

Description

This variable specifies to the tool how it is to write combinational logic and sequential logic. When you set the value of this variable as *true*, the compiler does the following:

- Writes combinational logic as technology-independent Boolean equations
- Writes sequential logic as technology-independent wait statements.

If you set the value of this variable to *false* (the default value), the tool writes all logic as technology-specific netlists.

Feedback

Note that if you define a bit type, make sure that Boolean equations are defined for the bit type.

vhdlout_follow_vector_direction

Specifies how the tool is to use the original range direction when it writes out an array.

Data Types

Boolean

Default false

Description

The *vhdlout_follow_vector_direction* variable defines the way in which the tool is to use the original range direction when it writes out an array.

To achieve this result, do the following:

- Set the value of this variable as *true* (the default value)
- Set the value of the *vhdlout_single_bit* variable to the type mode of *VECTOR*, not the ascending range.

For example, when you set the value of the *vhdlout_follow_vector_direction* variable as *true* (the default value), the HDL Compiler writes out the correct array range direction, such as 10 down to 0, if the original is 10 down to 0.

Note that if *vhdlout_follow_vector_direction* is set to *true*, the tool automatically sets the type mode to *VECTOR*.

See Also

- vhdlout_bit_type
- vhdlout_bit_vector_type
- vhdlout_preserve_hierarchical_types
- vhdlout_single_bit

vhdlout_lower_design_vector

Determines the way in which the *write -f vhdl* command writes out ports on lower-level designs

Data Types

Boolean

Feedback

Default true

Description

The *vhdlout_lower_design_vector* variable determines the way in which the *write -f vhdl* command writes out ports on lower-level designs. A lower-level design is instantiated by any of the designs being written out. The *vhdlout_top_design_vector* variable controls ports on top-level designs.

When the value of this variable is set to *false*, all ports on lower-level designs are written with their original data types. The value of *false* affects only designs read in VHDL format.

When set to *true* (the default) all ports on lower-level designs are written as bused ports, which means the ports keep their names and are not bit-blasted. The ports are written with the type you define using the *vhdlout_bit_vector_type* variable or, for single-bit ports, using the *vhdlout_bit_type* variable. This setting often results in the most efficient description for simulation. Bus ranges are always in ascending order beginning with 0, regardless of the range description in the original VHDL. If you set this variable as *true*, you must ensure that *vhdlout_bit_vector_type* is defined as an array type whose elements are of *vhdlout_bit_type*.

Note that you cannot set the value of the *vhdlout_lower_design_vector* variable to *false*, if the value of the *vhdlout_top_design_vector* variable is set to *true*.

Also note that if *vhdlout_follow_vector_direction* is set to*true*, the value of the *vhdlout_lower_design_vector* variable is automatically set to *true*.

See Also

- write
- vhdlout_bit_type
- vhdlout_bit_vector_type
- vhdlout_follow_vector_direction
- vhdlout_top_design_vector

---

vhdlout_one_name

Determines the literal name for constant bit value 1 in a design written in VHDL.

Data Types

string

Default '1'

Feedback

Description

The value you specify for the *vhdlout_one_name* variable determines the literal name for the constant bit value 1 in a design written in VHDL.

This variable is useful when your design methodology is based on a more general logic value than BIT. Use this variable with the *vhdlout_bit_type* variable.

See Also

- vhdlout_bit_type
- vhdlout_bit_vector_type
- vhdlout_three_state_name
- vhdlout_zero_name

---

vhdlout_package_naming_style

Determines the name the tool is to use for the type conversion packages written out by the VHDL writer (VHDLout).

Data Types

string

Default CONV_PACK_%d

Description

The *vhdlout_package_naming_style* variable determines the name the tool is to use for the type conversion packages written out by the VHDL writer (VHDLout). The value for this variable can contain a string made up of letters, digits, and underscores. The string *%d* is replaced by the name of the current design (*%d* and *%D* are synonymous), as shown as the default value. The tool gives unique names to all conversion packages that are written out.

The tool verifies that the generated name is a valid VHDL identifier and that it is a unique name. If the name is invalid, an error occurs.

---

vhdlout_preserve_hierarchical_types

Affects the way in which the *write -f vhdl* command writes out ports on lower-level designs

Data Types

string

Default VECTOR

Description

The *vhdlout_preserve_hierarchical_types* variable affects the way in which the *write -f vhdl* command writes out ports on lower-level designs. A lower-level design is instantiated by any of the designs being written out. Another variable, *vhdlout_single_bit*, controls ports on top-level designs.

Valid values are as follows:

USER

All ports on lower-level designs are written with their original data types. The value of *USER* affects only designs that are read in VHDL format.

VECTOR

All ports on lower-level designs are written with their ports bused, which means that ports keep their names and are not bit-blasted. The ports are written with the type defined by *vhdlout_bit_vector_type* or, for single-bit ports, by *vhdlout_bit_type*. This setting of *VECTOR* (the default value) often results in the most efficient description for simulation. Bus ranges are always in ascending order, beginning with 0, regardless of the range description in the original VHDL. If you use *VECTOR*, you must ensure that the value of the *vhdlout_bit_vector_type* variable is an array type whose elements are defined by the *vhdlout_bit_type* variable.

BIT

All typed ports are bit-blasted, which means that a port that is *n* bits wide is written to the VHDL file as *n* separate ports. Each port is given a type as defined by the *vhdlout_bit_type* variable.

Note that if *vhdlout_single_bit* = *BIT*, the tool ignores *vhdlout_preserve_hierarchical_types* and writes out the entire design hierarchy as bit-blasted. Also, *vhdlout_preserve_hierarchical_types* cannot take on a higher value than the current value defined for the *vhdlout_single_bit* variable. The descending order is *USER*, *VECTOR*, and *BIT*. Thus, the combination *vhdlout_single_bit* = *VECTOR* and *vhdlout_preserve_hierarchical_types* = *USER* is not possible.

Also note that if *vhdlout_follow_vector_direction* = *true*, the type mode is automatically set to VECTOR.

See Also

- write
- vhdlout_bit_type

Feedback

- vhdlout_bit_vector_type
- vhdlout_follow_vector_direction
- vhdlout_single_bit

vhdlout_separate_scan_in

Affects the way in which the *write -f vhdl* command writes out the scan chain in VHDL.

Data Types

string

Default false

Description

The *vhdlout_separate_scan_in* variable affects the way in which the *write -f vhdl* command writes out the scan chain in VHDL.

By default, the tool writes out the scan chain in the same file as the design. The scan chain is not visible in the testbench, and, therefore, parallel loading is not possible.

When the value is set to *true*, the tool writes out the scan chain as a package to a separate file. The design must be written out in a single hierarchical file, and it must contain the scan-chain package, before simulation or synthesis takes place.

You receive a message only when the scan-chain package is written successfully. Otherwise, no message appears.

See Also

- write

vhdlout_single_bit

Affects the way in which the *write -f vhdl* command writes out ports on the top-level design.

Data Types

string

Default USER

Description

This variable affects the way in which the *write -f vhdl* command writes out ports on the top-level design.

Feedback

Valid values are as follows:

USER

All ports on the top-level design are written out with their original data types. The *USER* value affects only designs that are read in VHDL format.

VECTOR

All ports on the top-level design are written with their ports bused, which means that ports keep their names and are not bit-blasted. The ports are written with the type defined by the *vhdlout_bit_vector_type* variable, or for single-bit ports, the *vhdlout_bit_type* variable. Lower-level design ports are controlled by the *vhdlout_preserve_hierarchical_types* variable. (A design is lower-level if it is instantiated by any of the designs being written out.)

Bus ranges are always in ascending order, beginning with 0, regardless of the range description in the original VHDL. If you use *VECTOR*, you must ensure that the value of the *vhdlout_bit_vector_type* variable is an array type whose elements are defined by the *vhdlout_bit_type* variable.

BIT

All typed ports are bit-blasted, which means that a port that is *n* bits wide is written to the VHDL file as *n* separate ports. Each port is given a type, as defined by the *vhdlout_bit_type* variable.

Note that if *vhdlout_follow_vector_direction* = *true*, the type mode is automatically set to VECTOR.

See Also

- write
- vhdlout_bit_type
- vhdlout_bit_vector_type
- vhdlout_preserve_hierarchical_types

---

vhdlout_target_simulator

Names the target simulator to which the tool writes the VHDL file.

Data Types

string

Default ""

Feedback

Description

The *vhdlout_target_simulator* variable names the target simulator to which the tool is to write the VHDL file. The only valid value is *xp*. The default is an empty string.

If you set the value of *vhdlout_target_simulator* as *xp*, be sure also to specify the *vhdlout_use_packages* variable to include the following instructions (or the written description does not analyze correctly):

SYNOPSYS.attributes.all

See Also

- vhdlout_use_packages

---

vhdlout_three_state_name

Names the high-impedance bit value used for three-state device values.

Data Types

string

Default 'Z'

Description

This variable names the high-impedance bit value used for three-state device values. The default value is *Z*.

See Also

- vhdlout_bit_type
- vhdlout_bit_vector_type
- vhdlout_one_name
- vhdlout_zero_name

---

vhdlout_three_state_res_func

Names a user-supplied three-state resolution function that must be in one of the packages specified by the *vhdlout_use_packages* variable.

Data Types

string

Feedback

Default ""

Description

The *vhdlout_three_state_res_func* variable names a user-supplied three-state resolution function that must be in one of the packages specified by the *vhdlout_use_packages* variable.

By default, the tool writes out (if needed) a default three-state resolution function. The default three-state resolution function drives a signal to "unknown," if the signal is driven more than once by logic 0 or logic 1.

See Also

- write
- vhdlout_use_packages
- vhdlout_wired_and_res_func
- vhdlout_wired_or_res_func

---

vhdlout_top_configuration_arch_name

Determines the name of the outside architecture, depending on the value you defined for the *vhdlout_write_top_configuration* variable, and causes the VHDL writer (VHDLout) to write out a configuration statement.

Data Types

string

Default A

Description

The value you define for the *vhdlout_top_configuration_arch_name* variable determines the name of the outside architecture, depending on the value you defined for the *vhdlout_write_top_configuration* variable, and causes the VHDL writer (VHDLout) to write out a configuration statement.

For example, assume the top-level design being written has an entity named *Ten* and architecture named *Tar*. The configuration statement binds *Ten*, *Tar*, and an outside entity named *Oen* with an outside architecture named *Oar* to instantiate *Ten* as a component. You set the variable as shown in the following example:

set_app_var vhdlout_top_configuration_arch_name {Oar}

An example of an outside architecture is a testbench that you provide to drive simulation with test vectors.

Feedback

See Also

- write
- vhdlout_top_configuration_entity_name
- vhdlout_top_configuration_name
- vhdlout_write_top_configuration

---

vhdlout_top_configuration_entity_name

Determines the name of the outside entity, depending on the value you defined for the *vhdlout_write_top_configuration* variable, and causes the VHDL writer (VHDLout) to write out a configuration statement.

Data Types

string

Default E

Description

The value you specify for the *vhdlout_top_configuration_entity_name* variable determines the name of the outside entity, depending on the value you defined for the *vhdlout_write_top_configuration* variable, and causes the VHDL writer (VHDLout) to write out a configuration statement.

For example, assume the top-level design being written has an entity named *Ten* and an architecture named *Tar*. The configuration statement binds *Ten*, *Tar*, and an outside entity named *Oen* with an architecture named *Oar*, to instantiate *Ten* as a component. You set the variable as shown in the following example:

set_app_var vhdlout_top_configuration_entity_name {Oen}

An example of an outside architecture is a testbench that you provide to drive simulation with test vectors.

See Also

- write
- vhdlout_top_configuration_arch_name
- vhdlout_top_configuration_name
- vhdlout_write_top_configuration

Feedback

vhdlout_top_configuration_name

Determines the name of the configuration statement the *write -f vhdl* command writes out, when the *vhdlout_write_top_configuration* variable is set to *true*.

Data Types

string

Default CFG_TB_E

Description

The value you specify for the *vhdlout_top_configuration_name* variable determines the name of the configuration statement the *write -f vhdl* command writes out, when the *vhdlout_write_top_configuration* variable is set to *true*.

See Also

- write
- vhdlout_top_configuration_entity_name
- vhdlout_write_top_configuration

vhdlout_top_design_vector

Affects the way in which the *write -f vhdl* command writes out ports on the top-level design.

Data Types

Boolean

Default false

Description

The value you specify for the *vhdlout_top_design_vector* variable determines the way in which the *write -f vhdl* command writes out ports on top-level designs.

By default, all ports on top-level designs are written out with their original data types.

If you set the value of this variable to *true*, all ports on top-level designs are written as bused ports, which means the ports keep their names and are not bit-blasted. The ports are written with the type you define using the *vhdlout_bit_vector_type* variable or, for single-bit ports, using the *vhdlout_bit_type* variable.

Bus ranges are always in ascending order beginning with 0, regardless of the range description in the original VHDL. If you set this variable to *true*, you must ensure that *vhdlout_bit_vector_type* is an array type whose elements are of *vhdlout_bit_type*.

Feedback

Note that if *vhdlout_follow_vector_direction* is set to *true*, the value of the *vhdlout_top_design_vector* variable is automatically set to *true*.

See Also

- write
- vhdlout_bit_type
- vhdlout_bit_vector_type
- vhdlout_follow_vector_direction
- vhdlout_lower_design_vector

vhdlout_unconnected_pin_prefix

Affects the way in which the *write -f vhdl* command writes out unconnected pin names.

Data Types

string

Default n

Description

The *vhdlout_unconnected_pin_prefix* variable affects the way in which the *write -f vhdl* command writes out unconnected pin names. You can set the value to any string, but to work with the *change_names* command's default naming prefix, the recommended value for this variable is SYNOPSYS_UNCONNECTED_.

See Also

- write
- vhdlout_dont_create_dummy_nets

vhdlout_unknown_name

Specifies the value the tool is to use to drive a signal to the "unknown" state.

Data Types

string

Default 'X'

Feedback

Description

The *vhdlout_unknown_name* variable specifies the value the tool is to use to drive a signal to the "unknown" state.

See Also

- vhdlout_one_name
- vhdlout_three_state_name
- vhdlout_zero_name

---

vhdlout_upcase

This variable is obsolete.

---

vhdlout_use_packages

Instructs the *write -f vhdl* command to write into the VHDL file a "use clause" that contains a list of package names for each of the packages described in this man page, for all entities.

Data Types

list

Default IEEE.std_logic_1164

Description

This variable instructs the *write -f vhdl* command to write into the VHDL file a use clause containing a list of package names, for each of the packages this man page describes, for all entities.

The *write -f vhdl* command also writes out clauses called "library clauses" as needed. If this variable is set to an empty list (""), the variable has no effect on the *write -f vhdl* command, and library clauses are not written out.

The process by which each package is printed out depends on the number of dots (.) in the package name, as follows:

- `No dots prepends "work." and appends ".all" to the name.`
- `One dot appends ".all" to the name.`
- `Two dots simply prints the package name.`

Feedback

For example:

```
types          ===> library work   use work.types.all

my.types       ===> library my     use my.types.all

my.types.and   ===> library my     use my.types.and
```

See Also

- write

---

vhdlout_wired_and_res_func

Specifies the name of a "wired and" resolution function.

Data Types

string

Default ""

Description

The *vhdlout_wired_and_res_func* variable specifies the name of a "wired and" resolution function. The name of this user-supplied function must be included in one of the packages the *vhdlout_use_packages* variable specifies. If this variable is set to an empty string, a default "wired and" resolution function is written out, if needed.

See Also

- vhdlout_three_state_res_func
- vhdlout_use_packages
- vhdlout_wired_or_res_func

---

vhdlout_wired_or_res_func

Specifies the name of a "wired or" resolution function.

Data Types

string

Default ""

Feedback

Description

The *vhdlout_wired_or_res_func* variable specifies the name of a "wired or" resolution function. The name of this user-supplied function must be included in one of the packages the *vhdlout_use_packages* variable specifies. If this variable is set to an empty string, a default "wired or" resolution function is written out, if needed.

See Also

- vhdlout_three_state_res_func
- vhdlout_use_packages
- vhdlout_wired_and_res_func

---

vhdlout_write_architecture

Instructs the *write -format vhdl* command to write out architecture declarations.

Data Types

Boolean

Default true

Description

The *vhdlout_write_architecture* variable instructs the *write -format vhdl* command to write out architecture declarations. When set to *false*, no architecture declarations are written.

Use this variable with the *vhdlout_write_entity* and *vhdlout_write_top_configuration* variables to control the information the *write -format vhdl* command writes out.

See Also

- write
- vhdlout_write_entity
- vhdlout_write_top_configuration

---

vhdlout_write_components

Instructs the *write -format vhdl* command to write out component declarations for cells mapped to a technology library.

Data Types

Boolean

Feedback

Default true

Description

The *vhdlout_write_components* variable instructs the *write -format vhdl* command to write out component declarations for cells mapped to a technology library.

When set to *false*, no component declarations are written. The VHDL format requires component declarations. If you set this variable to *false*, make sure that the *vhdlout_use_packages* variable includes a package containing the necessary component declarations.

See Also

- write
- vhdlout_use_packages

---

vhdlout_write_entity

Instructs the *write -format vhdl* command to write out entity declarations.

Data Types

Boolean

Default true

Description

The *vhdlout_write_entity* variable instructs the *write -format vhdl* command to write out entity declarations.

When set to *false*, no entity declarations are written.

Use this variable with the *vhdlout_write_architecture* and *vhdlout_write_top_configuration* variables to control the information the *write -format vhdl* command writes out.

See Also

- write
- vhdlout_write_architecture
- vhdlout_write_top_configuration

Feedback

vhdlout_write_top_configuration

Instructs the *write -format vhdl* command to write out a configuration statement, if necessary, such as when ports on the top-level design are written as vectors instead of user types.

Data Types

Boolean

Default false

Description

The *vhdlout_write_top_configuration* variable to instructs the *write -format vhdl* command to write out a configuration statement, if necessary, such as when ports on the top-level design are written as vectors instead of user types. The port map of the configuration statement contains calls to type conversion functions.

For more information, see the man pages for the *vhdlout_single_bit* and *vhdlout_preserve_hierarchical_types* variables.

By default, the tool does not write out a configuration statement.

The configuration statement binds the (written out) top-level design to an entity and architecture outside of the .db file. The outside architecture is assumed to instantiate as a component the written top-level design. An example of an outside architecture is a user-supplied testbench that drives simulation with test vectors.

For example, if the testbench uses user types, but the top-level design's ports are written as vectors, the ports do not interface to the testbench directly because of type mismatches. The use of type conversions in the configuration statement solves this problem. The *write -format vhdl* command does not write a configuration statement for bit-blasted designs.

Inout ports cannot be handled by conversion functions, because there is no way of determining whether to convert for a sink (output) or for a source (input). When *write -format vhdl* encounters an inout port, it checks the netlist. If the port functions as a true bidirectional signal, the tool issues error message VHDL-17. If the inout port functions only as a sink or as a source, *write -format vhdl* creates the correct type conversion.

You can use this variable with the *vhdlout_write_architecture* and *vhdlout_write_entity* variables to control the information *write -format vhdl* writes out.

See Also

- write
- vhdlout_preserve_hierarchical_types

Feedback

- vhdlout_single_bit
- vhdlout_top_configuration_arch_name
- vhdlout_top_configuration_entity_name
- vhdlout_top_configuration_name
- vhdlout_write_architecture
- vhdlout_write_entity

vhdlout_zero_name

Determines the literal name for constant bit value 0 in a design written in VHDL.

Data Types

string

Default '0'

Description

The *vhdlout_zero_name* variable determines the literal name for the constant bit value 0 in a design written in VHDL.

This variable is useful when your design methodology is based on a more general logic value than BIT. Use this variable with the *vhdlout_bit_type* variable.

See Also

- write
- vhdlout_bit_type
- vhdlout_bit_vector_type
- vhdlout_one_name
- vhdlout_three_state_name

view_analyze_file_suffix

Specifies, in a list of file extensions, the files shown in the File/Analyze dialog box of Design Analyzer.

Data Types

string

Default v vhd vhdl in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Specifies, in a list of file extensions, the files shown in the File/Analyze dialog box of Design Analyzer. The default value is {*v*, *vhd*, *vhdl*}.

To determine the current value of this variable, type *printvar view_analyze_file_suffix*. For a list of all *view* variables and their current values, type *print_variable_group view*.

---

view_arch_types

Sets the contents of the architecture option menu. Contains a list of host machine architectures you can use for background jobs from the Design Analyzer viewer.

Data Types

list

Default sparcOS5 hpux10 rs6000 sgimips in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable sets the contents of the architecture option menu. It contains a list of host machine architectures you can use for background jobs from the Design Analyzer viewer.

---

view_background

Specifies the background color of the Design Analyzer viewer.

Data Types

string

Default black in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Feedback

Description

Specifies the background color of the Design Analyzer viewer. Valid settings are *white* (the default) and *black*.

To determine the current value of this variable, type *printvar view_background*. For a list of all *view* variables and their current values, type *print_variable_group view*.

---

view_cache_images

Specifies to Design Analyzer that the tool is to cache bitmaps for fast schematic drawing.

Data Types

string

Default true in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Specifies that the tool is to cache bitmaps for fast schematic drawing. The default is *true*.

To determine the current value of this variable, type *printvar view_cache_images*. For a list of all *view* variables and their current values, type *print_variable_group view*.

---

view_command_log_file

Names a file and its location that is to contain all text written to the Design Analyzer Command window.

Data Types

string

Default Variable is not supported in DC Explorer

Description

Names a file and its location that is to contain all text written to the Design Analyzer Command window. The default is to set this variable.

To determine the current value of this variable, use *printvar view_command_log_file*. For a list of all *view* variables and their current values, type *print_variable_group view*.

Feedback

See Also

- command_log_file
- view_log_file

---

view_command_win_max_lines

Contains the maximum number of lines to be saved in the Design Analyzer command window.

Data Types

integer

Default 1000 in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Contains the maximum number of lines to be saved in the Design Analyzer command window. When the number of lines of output added to the command window exceed the number this variable specifies, the older lines at the top of the list are removed.

This variable should be set to 1000 or more. Values up to tens of thousands are also useful.

To determine the current value of this variable, type *printvar view_command_win_max_lines*. For a list of all *view* variables and their current values, type *print_variable_group view*.

---

view_dialogs_modal

Requires that the question and error dialogs in Design Analyzer be confirmed, before you can continue entering commands.

Data Types

Boolean

Default true in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Feedback

Description

Requires that the question and error dialogs in Design Analyzer be confirmed, before you can continue entering commands.

To determine the current value of this variable, use *printvar view_dialogs_modal*. For a list of all *view* variables and their current values, type the *print_variable_group view*.

---

view_disable_cursor_warping

Causes the cursor to be automatically "warped" (moved). When *false*, the cursor is automatically "warped" (or moved) to dialog boxes.

Data Types

Boolean

Default true in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Causes the cursor to be automatically "warped" (moved). When *false*, the cursor is automatically "warped" (or moved) to dialog boxes. The default is *true*.

To determine the current value of this variable, use *printvar view_disable_cursor_warping*. For a list of all *view* variables and their current values, type *print_variable_group view*.

---

view_disable_error_windows

Instructs Design Analyzer not to post the error windows when errors occur.

Data Types

Boolean

Default false in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Instructs Design Analyzer not to post the error windows when errors occur. The default is *false*.

To determine the current value of this variable, type *printvar view_disable_error_windows*. For a list of all *view* variables and their current values, type *print_variable_group view*.

---

view_disable_output

Disables output to the Design Analyzer command window.

Data Types

Boolean

Default false in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Disables output to the Design Analyzer command window, when set to *true*. This variable is useful when you run Design Analyzer over slow networks, such as telephone lines. The default value is *false*.

To determine the current value of this variable, type *printvar view_disable_output*. For a list of all *view* variables and their current values, type *print_variable_group view*.

---

view_error_window_count

Specifies the maximum number of errors Design Analyzer reports for a command.

Data Types

integer

Default 6 in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Specifies the maximum number of errors Design Analyzer reports for a command. The default value is *6*. If more than the specified number of errors occurs, you are informed that you can see additional errors in the command window. The error window is suppressed until the end of the command.

To display all errors, set this variable to 0. To display no errors, set this variable to a negative number or set the *view_disable_error_windows* variable to *true*.

To determine the current value of this variable, type *printvar view_error_window_count*. For a list of all *view* variables and their current values, type *print_variable_group view*.

See Also

- view_disable_error_windows

---

view_execute_script_suffix

Displays only files with the stated suffixes, from directories you select in the Execute Script option window of the Setup menu of Design Analyzer.

Data Types

list

Default Variable is not supported in DC Explorer

Description

Displays only files with the stated suffixes, from directories you select in the Execute Script option window of the Setup menu of Design Analyzer.

To determine the current value of this variable, type *printvar view_execute_script_suffix*. For a list of all *suffix* variables and their current values, type *print_variable_group suffix*.

---

view_info_search_cmd

Invokes, if set, the online information viewer through the optional menu item On-Line Information.

Data Types

string

Default ""

Description

Invokes, if set, the online information viewer through the optional menu item On-Line Information. Set the value of this variable to the UNIX path name to the online information viewer.

To determine the current value of this variable, type *printvar view_info_search_cmd*. For a list of all *view* variables and their current values, type *print_variable_group view*.

Feedback

view_log_file

Specifies the file in which the tool stores events that occur in the viewer.

Data Types

string

Default "" in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Specifies the file in which the tool stores events that occur in the viewer. This variable is useful for error reporting. You can execute this variable with the Execute Script option of the Setup menu, or insert the file, using the *include* command in Design Analyzer.

To determine the current value of this variable, type *printvar view_log_file*. For a list of all *view* variables and their current values, type *print_variable_group view*.

view_on_line_doc_cmd

Invokes, if set, the online documentation viewer, through the optional menu item On-Line Documentation.

Data Types

string

Default ""

Description

Invokes, if set, the online documentation viewer, through the optional menu item On-Line Documentation. Set the value of this variable to the UNIX command to invoke the online documentation view: for example, /vobs/snps/synopsys/sold.

To determine the current value of this variable, type *printvar view_on_line_doc_cmd*. For a list of all *view* variables and their current values, type *print_variable_group view*.

view_read_file_suffix

Displays only files with the stated suffixes, from directories you select with the Read option of the File menu of Design Analyzer.

Feedback

Data Types

list

Default db gdb sdb edif eqn fnc lsi mif NET pla st tdl v vhd vhdl xnf

Description

Displays only files with the stated suffixes, from directories you select with the Read option of the File menu of Design Analyzer.

To determine the current value of this variable, type *printvar view_read_file_suffix*. For a list of all *suffix* variables and their current values, type *print_variable_group suffix*.

---

view_script_submenu_items

Allows users to add to the Design Analyzer Setup pulldown menu valid items to invoke user scripts.

Data Types

string

Default "DA to SGE Transfer" write_sge in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Allows users to add to the Design Analyzer Setup pulldown menu valid items to invoke user scripts. The variable should contain a list of strings, grouped into pairs. The first member of the pair is the text that will appear in the submenu. The second member is the string that gets sent to the dc_shell command line for execution. You can use any valid dc_shell command sequence.

For example,

view_script_submenu_items = "{"List", "list_instances", "ls", "sh ls -lR", "Update", "include update.dcsh"}"

creates a submenu under the Scripts menu item on the Setup pulldown menu. The submenu contains the strings List, ls, and Update. Selecting one of these entries executes the commands *list_instances*, *sh ls -lR*, or *include update.dcsh*, respectively.

The tool reads this variable only at startup time, so any changes after the Design Analyzer is initialized are not reflected.

To determine the current value of this variable, type *list view_script_submenu_items*. For a list of all *view* variables and their current values, type *print_variable_group view*.

---

view_set_selecting_color

Specifies the color to use for selecting and zooming.

Data Types

string

Description

Specifies the color to use for selecting and zooming. You can set this variable in the *.synopsys_dc.setup* initialization file. Any color in */usr/lib/X11/rgb.txt* is valid.

To determine the current value of this variable, type *printvar view_set_selecting_color*. For a list of all *view* variables and their current values, type *print_variable_group view*.

---

view_tools_menu_items

Permits partial configuration of the Tools pulldown menu to add a new menu item for invoking user scripts.

Data Types

string

Default "" in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Permits partial configuration of the Tools pulldown menu to add a new menu item for invoking user scripts. This .synopsys_dc.setup file variable contains a list of strings, grouped into pairs. The first member of the pair is the text that appears in the submenu. The second member is the string that is sent to the dc_shell command line for execution. You can use any valid dc_shell command sequence.

For example,

view_tools_menu_items = "{"List", "list_instances", "ls", "sh ls -lR", "Update", "include update.dcsh"}"

adds three more items to the Tools menu: List, ls, and Update. Selecting one of these entries executes one of the commands. For instance, if you selected *update*, the *include update.dcsh* command would be executed.

The tool reads this variable only at start-up time. Any changes after the Design Analyzer is initialized are not reflected.

To determine the current value of this variable, type *printvar view_tools_menu_items*. For a list of all *view* variables and their current values, type *print_variable_group view*.

---

view_use_small_cursor

Specifies to the tool that the X display is to support only 16 x 16-bit map size cursors.

Data Types

string

Default true in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

Specifies to the tool that the X display is to support only 16 x 16-bit map size cursors. To achieve this result, set the value of this variable to *true*. The default is *""*.

To determine the current value of this variable, type *printvar view_use_small_cursor*. For a list of all *view* variables and their current values, type *print_variable_group view*.

---

view_use_x_routines

Enables the use of internal arc-drawing routines (instead of X routines).

Data Types

Boolean

Default true in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Feedback

Description

Enables the use of internal arc-drawing routines (instead of X routines). If there is a math coprocessor chip on the same machine that the X server is on, X arc-drawing routines are faster. Otherwise, internal arc-drawing routines are faster. The default value is *true*.

To determine the current value of this variable, type *printvar view_use_x_routines*. For a list of all *view* variables and their current values, type *print_variable_group view*.

---

view_write_file_suffix

Displays only files with the stated suffixes, from directories you select with the Save As option of the File menu of Design Analyzer.

Data Types

list

Default gdb db sdb do edif eqn fnc lsi NET neted pla st tdl v vhd vhdl xnf

Description

Displays only files with the stated suffixes, from directories you select with the Save As option of the File menu of Design Analyzer.

To determine the current value of this variable, type *printvar view_write_file_suffix*. For a list of all *suffix* variables and their current values, type *print_variable_group suffix*.

W

---

wildcards

Describes supported wildcard characters and ways in which they can be escaped.

Description

The following characters are supported as wildcards:

- Asterisks (*) substitute for a string of characters of any length.
- Question marks (?) substitute for a single character.

The following commands support wildcard characters:

```
get_cells
get_clocks
get_designs
get_lib_cells
```

Feedback

```
get_lib_pins
get_libs
get_multibits
get_nets
get_pins
get_ports
get_references
list_designs
list_instances
list_libs
trace_nets
untrace_nets
```

In addition to these commands, commands that perform an implicit get also support the wildcarding feature.

Escaping Wildcards

Wildcard characters must be escaped using double backslashes (\\\\) to remove their special regular expression meaning. See the EXAMPLES section for more information.

Escaping Escape Character (\\\\)

This is similar to that of escaping wildcard characters, but needs one escape character each to escape the escape character. See the EXAMPLES section for more information.

Examples

The following are examples of using wildcard characters.

Using Wildcards

The following example gets all nets in the current design that are prefixed by in and followed by any two characters:

```
prompt> get_nets in??
{"in11" "in21"}
```

The following example gets all cells in the current design that are prefixed by U and followed by a string of characters of any length:

```
prompt> get_cells U*
{"U1" "U2" "U3" "U4"}
```

Escaping Wildcards

The following are examples of escaping wildcard characters.

The following example gets design test?1 in the system.

```
prompt> get_designs {test\\\\?1}
{"test?1"}
```

Feedback

The same example can be written using the Tcl *list* command:

```
prompt> get_designs [list {test\\?1}]
{"test?1"}
```

If neither curly braces nor the *list* command is used, the syntax is as follows:

```
prompt> get_designs test\\\\\\\\?1
{"test?1"}
```

Escaping Escape Character (\\\\)

The following are examples of escaping an escape character.

The following example finds design test\\1 in the system.

```
prompt> get_designs {test\\\\\\\\1}
{"test\\1"}
```

The same above example can written using the Tcl *list* command:

```
prompt> get_designs [list {test\\\\1}]
{"test\\1"}
```

If neither curly braces nor the *list* command is used, the syntax is as follows:

```
prompt> get_designs test\\\\\\\\\\\\\\\\1
{"test\\1"}
```

---

write_converted_tf_syntax

Controls whether the *write_mw_lib_files* command writes out the converted rule tables when the technology file uses the alternative syntax for area-based fat metal contact rules and fat metal extension contact rules.

Data Types

Boolean

Default false

Description

This variable controls the syntax used by the *write_mw_lib_files* command for the area-based fat metal contact rules and fat metal extension contact rules when these rules are defined in the technology file using the alternative syntax.

The alternative syntax defines these rules by using the enclosureTbl and minCutsTbl tables in the Layer section of the technology file. Internally, the tool converts these rules to ContactCode sections and via rules in the Layer section.

Feedback

By default, the *write_mw_lib_files* command writes these rules using the syntax from the input technology file. If you set this variable to *true*, the *write_mw_lib_files* command writes these rules using the internally-derived syntax (the Contact Code sections and via rules in the Layer sections).

For details about the technology file syntax for these rules, see the *IC Compiler Technology File and Routing Rules Reference Manual*.

See Also

- write_mw_lib_files

---

write_name_nets_same_as_ports

Specifies to the tool that nets are to receive the same names as the ports to which the nets are connected.

Data Types

Boolean

Default false

Description

This variable specifies to the tool that nets are to receive the same names as the ports to which the nets are connected. This variable affects nets in design descriptions written in EDIF, LSI, or TDL format. (Other nets might be renamed to avoid creating shorts.)

Note that in the EDIF format, even if the *edifout_power_and_ground_representation* variable is set to *port*, the power and ground nets will not be named the same as the power and ground ports that are written in the interface construct of each cell construct.

Net names in the design database are unchanged; that is, only the file being written out is affected and not the original design data.

---

write_sdc_output_lumped_net_capacitance

Determines whether or not the *write_sdc* and *write_script* commands output net loads.

Data Types

Boolean

Default true

Feedback

Description

This variable instructs the *write_sdc* and *write_script* commands to output net loads when set to *true*.

When set to *false*, *write_sdc* and *write_script* do not output net loads. The net loads are output through *set_load* command statements during *write_sdc* and *write_script*.

By default all net loads are output during *write_sdc* and *write_script* execution.

See Also

- set_load
- write_sdc
- write_script

---

write_sdc_output_net_resistance

Determines whether or not the *write_sdc* and *write_script* commands output net resistance.

Data Types

Boolean

Default true

Description

This variable instructs the *write_sdc* and *write_script* commands to output net resistance. When set to false, *write_sdc* and *write_script* do not output net resistance. The net resistances are output through *set_resistance* statements during *write_sdc* and *write_script*.

By default, all net resistances are output during *write_sdc* and *write_script* execution.

See Also

- set_resistance
- write_sdc
- write_script

---

write_test_formats

Specifies the test vector formats recognized and created by the *write_test* command.

Feedback

Data Types

list

Default synopsys tssi_ascii tds verilog vhdl wgl

Description

This variable specifies the test vector formats recognized and created by the *write_test* command.

See Also

- write_test

---

write_test_include_scan_cell_info

Specifies to the *write_test* command to include in the vector files scan-chain, cell, and inversion information for vector formats.

Data Types

string

Default true

Description

This variable instructs the *write_test* command to include in the vector files scan-chain, cell, and inversion information for vector formats.

For a simulator to execute a parallel load of scan chains, it is necessary to establish the instance names of all elements in the scan registers and to know whether there is inversion between each element and the scan-data input or output. This is the information that *write_test* includes by default.

To exclude the information from the vector files, set the *write_test_include_scan_cell_info* variable to *false*.

See Also

- write_test

---

write_test_input_dont_care_value

Controls the logic value that the *write_test* command outputs when you have an input with a don't care condition.

Feedback

Data Types

string

Default X

Description

This variable controls the logic value that the *write_test* command outputs when you have an input with a don't care condition.

For example, in a design with multiple scan chains of unequal lengths, the serial input streams for the shorter chains need to be padded with arbitrary logic values.

This variable can have the value *1*, *0*, *X* or *x*. These values correspond to a don't care condition of logic 1, logic 0, or logic don't care, respectively. The default is *don't care* (*X*).

For test equipment or test vector formats that do not support an input don't care value, use this variable to assign logic 1 or logic 0.

See Also

- write_test

---

write_test_max_cycles

Controls the automatic partitioning of long test sets across multiple files by specifying the maximum number of tester cycles any one vector file can contain.

Data Types

integer

Default 0

Description

This variable controls the automatic partitioning of long test sets across multiple files by specifying the maximum number of tester cycles any one vector file can contain. One tester cycle corresponds to one cycle of parallel (nonshift) operation of the device under test or to the shifting of scan data one bit in the scan chains.

Each vector file is self-contained; that is, the vectors within a specific file are independent of the state of the device under test and specifically do not require the vectors in other files to have been previously applied. This implies that *write_test_max_cycles* is used only for full-scan designs, and that test sets are not partitioned for partial scan designs. Additionally, the partitioning of the vector set has no effect on fault coverage.

Feedback

By default, this variable is set to *0*, which implies there are no limits. Thus, automatic file partitioning is not performed. Work with your ASIC vendors to determine an appropriate value.

See Also

- write_test_max_scan_patterns
- write_test_vector_file_naming_style

---

write_test_max_scan_patterns

Controls the automatic partitioning of long test sets across multiple files, by specifying the maximum number of scan test patterns any one vector file can contain.

Data Types

integer

Default 0

Description

This variable controls the automatic partitioning of long test sets across multiple files, by specifying the maximum number of scan test patterns any one vector file can contain. The application of a scan test pattern encompasses the serial loading and unloading of the scan chains and consumes many tester cycles. Typically this correlates directly to the number of scan test patterns the *create_test_patterns* command generates. For designs with multiple system clocks, however, it might be necessary to repeat a number of times the application of each scan test pattern.

Each vector file is self-contained; that is, the vectors within a specific file are independent of the state of the device under test and do not require the vectors in other files to have been previously applied. This implies that *write_test_max_scan_patterns* is used only for full-scan designs, and that test sets are not partitioned for partial-scan designs. Additionally, the partitioning of the vector set has no effect on fault coverage for full-scan designs.

By default, this variable is set to *0*, which implies there are no limits. Thus, automatic file partitioning is not performed. Work with your ASIC vendors to determine an appropriate value. Setting this variable to any value less than 2 results in no file partitioning.

See Also

- write_test_max_cycles
- write_test_vector_file_naming_style

Feedback

write_test_pattern_set_naming_style

Specifies how to name pattern sets when long test sets are partitioned across multiple files.

Data Types

string

Default TC_Syn_%d

Description

This variable specifies how to name pattern sets when long test sets are partitioned across multiple files. The pattern set name is a documentation aid and appears inside a comment header in the vector files.

The value of this variable is a format string containing one *%d* token that corresponds to the file number index. The file number index starts at 0 and, if the test set is partitioned, is incremented by 1 as additional files are needed.

write_test_round_timing_values

Specifies to the *write_test* command to round all timing values to the nearest integer.

Data Types

string

Default true

Description

This variable specifies to the *write_test* command to round all timing values to the nearest integer. The values are input delay, output strobe time, bidir delay, clock period, and clock edge times.

If you do not want the timing values to be rounded to the nearest integer, set the value of this variable to *false*.

When this variable is set to *false*, the maximum resolution of the timing values generated by *write_test* is 0.01. For finer resolution, change the time units (for example, from ns to ps).

See Also

- write_test

Feedback

write_test_scan_check_file_naming_style

Specifies how to name the file containing the vectors that test the scan chain logic.

Data Types

string

Default %s_schk.%s

Description

This variable specifies how to name the file containing the vectors that test the scan chain logic.

The value of this variable is a format string containing two *%s* tokens. The first token pertains to the base name of the file specified by the *-output* option of the *write_test* command, or the value defaults to the design name. The second token is a file extension based on the targeted vector format, such as .v, for Verilog formatted vectors.

See Also

- write_test
- write_test_max_cycles
- write_test_max_scan_patterns

write_test_vector_file_naming_style

Specifies how to name scan vector files when long test sets are partitioned across multiple files.

Data Types

string

Default %s_%d.%s

Description

This variable specifies how to name scan vector files when long test sets are partitioned across multiple files.

The value of this variable is a format string containing the *%s*, *%d*, and *%s* tokens, in this exact order. The first token, *%s*, pertains to the base name of the file specified by the *-output* option of the *write_test* command, or the value defaults to the design name. The second token, *%d*, corresponds to the file number index. The file number index starts at 0 and if the test set is partitioned, is incremented by 1 as additional files are needed.

The third token, *%s*, is a file extension based on the targeted vector format, such as .v for Verilog formatted vectors.

See Also

- write_test
- write_test_max_cycles
- write_test_max_scan_patterns

write_test_vhdlout

Determines whether the *write_test -format vhdl* command generates a VHDL test program in TEXTIO format.

Data Types

string

Default inline

Description

This variable specifies whether or not the *write_test -format vhdl* command generates a VHDL test program in TEXTIO format.

If you set the value of this variable as *textio*, the tool writes the VHDL test program in TEXTIO format. If you set the value to *inline* (the default), TEXTIO format is not used.

Compiling a large number of patterns (and placing them in memory) can sometimes cause swapping during simulation. Using TEXTIO format is helpful in such a situation. Both serial load and parallel load testbench generation are supported. To generate a parallel loadable testbench, use the *write_test -parallel* command.

See Also

- write_test

X

x11_set_cursor_background

Specifies background color of the cursor in the Design Analyzer menus and viewer.

Feedback

Data Types

string

Default "" in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

Specifies background color of the cursor in the Design Analyzer menus and viewer. This variable can be set in the *.synopsys_dc.setup* initialization file. Any color in */usr/lib/X11/rgb.txt* is valid.

To determine the current value of this variable, type *printvar x11_set_cursor_background*. For a list of all *view* variables and their current values, type *print_variable_group view*.

---

x11_set_cursor_foreground

Specifies foreground color of the cursor in the Design Analyzer menus and viewer.

Data Types

string

Default magenta

Description

Specifies foreground color of the cursor in the Design Analyzer menus and viewer. This variable can be set in the *.synopsys_dc.setup* initialization file. Any color in */usr/lib/X11/rgb.txt* is valid.

To determine the current value of this variable, type *printvar x11_set_cursor_foreground*. For a list of all *view* variables and their current values, type *print_variable_group view*.

---

x11_set_cursor_number

Specifies the cursor, from the standard X cursor font used by the Design Analyzer menus and viewer.

Data Types

integer

Default -1 in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

Specifies the cursor, from the standard X cursor font used by the Design Analyzer menus and viewer.

If this variable is not set, or is set to *-1* (the default), the cursor in the X background is also used for all windows and menus. Set this variable in the *.synopsys_dc.setup* initialization file.

To determine the current value of this variable, type *printvar x11_set_cursor_number*. For a list of all *view* variables and their current values, type *print_variable_group view*.

---

xt_filter_logic_constant_aggressors

Affects the behavior of the *report_noise*, *report_timing*, and *compile* commands with crosstalk or noise effect enabled. Specifies if logic constant nets should be considered as aggressors in crosstalk and static noise analysis and optimization.

Data Types

Boolean

Default true in Design Compiler non-topographical mode, Design Compiler topographical mode, Design Compiler NXT non-topographical mode, and Design Compiler NXT topographical mode

Variable is not supported in DC Explorer

Description

This variable affects the behavior and correlation of the *report_noise*, *report_timing*, and *compile* commands. By default, logic constant aggressors are not included in crosstalk or static noise analysis and optimization. Aggressor nets can be logic constant because they are connected to tie-off cells or because of case analysis. The *case_analysis_with_logic_constants* timing variable may also determine if an aggressor net is logic constant.

When set to *false*, logic constant aggressors are included in crosstalk or static noise analysis and optimization. To obtain the more accurate behavior of crosstalk timing and static noise analysis, set this variable to *true*.

Feedback

See Also

- compile
- report_timing
- case_analysis_with_logic_constants

xterm_executable

Specifies the path to an xterm program spawned to run Synopsys analysis tools (for example, RTL Analyzer).

Data Types

string

Default xterm in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

Variable is not supported in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Description

This variable specifies the path to an xterm program spawned to run Synopsys analysis tools (for example, RTL Analyzer).

If an xterm is not found in your path, set this variable to point to an xterm executable. For example, if your xterm is not in your $path, but is located at /usr/bin/X11/xterm, set this variable as follows:

```
prompt> set_app_var xterm_executable {/usr/bin/X11/xterm}
```

z

zrt_max_parallel_computations

Sets the maximum degree of parallelism in multicore Zroute.

Data Types

integer

Default Variable is not supported in Design Compiler non-topographical mode and Design Compiler NXT non-topographical mode

0 in Design Compiler topographical mode, DC Explorer, and Design Compiler NXT topographical mode

Group

Zroute

Description

This application variable specifies the degree of parallelism in multicore Zroute. Increasing parallelism results in reduced runtime.

When set to *0* (the default), the number of cores used for Zroute is specified directly by the value of the *set_host_options* command with the *-max_cores* option value.

When set to *1*, multicore Zroute is disabled.

When the value is smaller than the *-max_cores* value of the *set_host_options* command, the number of cores being used will not exceed this limit.

There is no benefit to setting the value to larger than *-max_cores*. The actual degree of parallelism (number of processes/threads) will not exceed the value specified here, but may not always be exactly the same. The tool derives the number internally to achieve best performance.

See Also

- set_host_options

2

Synthesis Attributes

This document describes the attributes supported by the Design Compiler tool.

a

acs_compile_attributes

Customizes the Automated Chip Synthesis compile strategy.

Description

Automated Chip Synthesis provides attributes that can customize the compile strategy. To set these attributes, use the *set_attribute* command to set the attribute on the reference design for the compile partition (or on the top-level design).

Each chip-level compile command (acs_compile_design, acs_refine_design, and acs_recompile_design) performs a two-step compile. The acs_compile_design and acs_recompile_design commands perform a full compile followed by a boundary compile. The acs_refine_design command performs an incremental compile followed by a boundary compile. The Automated Chip Synthesis compile attributes control both the effort and the strategy used in each of the steps.

These attributes affect the compile scripts generated by the *write_compile_script* command (and therefore the compile scripts generated by the chip-level compile commands). Some attributes add commands to the script before the compile command; others modify the compile command options.

Automated Chip Synthesis ignores these compile attributes if a custom compile strategy or custom compile script exists in the directory defined for the user_compile_strategy_script or user_override_script directories (default directory is $acs_work_dir/scripts/dest_dir).

The following attributes control the effort used in each compile step (full, incremental, and boundary):

```
• FullCompile
Valid values are none, low, medium, and high.
Default value is medium.
```

```
• IncrementalCompile
Valid values are none, low, medium, and high.
Default value is high.


• BoundaryCompile
Valid values are none, low, medium, high, and top.
Default value is top.
```

The following attributes modify the compile strategy for each compile step:

```
• OptimizationPriorities (all modes)
Valid values are area, area_timing, timing, and timing_area.
Default value is timing_area.


• CanFlatten (full compile mode only)
Valid values are true and false.
Default value is false.


• CompileVerify (full and incremental compile modes only)
Valid values are true and false.
Default value is false.


• HasArithmetic (full compile mode only)
Valid values are true and false.
Default value is false.
(Deprecated: You will get better QoR for datapathes with just
 UltraOptimization enabled.)


• PartitionDP (full compile mode only)
Valid values are duplicate, dont_split and false.
Default value is false.
(Deprecated: You will get better QoR for datapathes with just
 UltraOptimization enabled.)


• AutoUngroup (full compile mode only)
This is a composite attribute with one string value with the
syntax: <mode> [<numCells>]
Valid values for the required mode are area, delay and false.
Default value is false.
Valid values for the optional numCells are integer numbers >0.
Default value is unset.


• MaxArea (full and incremental compile modes only)
Valid values are floating point numbers greater than or equal to 0.0.
Default value is 0.0.
```

Feedback

• PreserveBoundaries (full and incremental compile modes only)
Valid values are true and false.
Default value is true.

• TestReadyCompile (full and incremental compile modes only)
Valid values are true and false.
Default value is false.

• UltraOptimization (all compile modes)
Valid values are true and false.
Default value is false.

For details about the impact of each attribute on the compile script, see the *Automated Chip Synthesis User Guide*.

See Also

- compile
- get_license
- remove_license
- set_cost_priority
- set_flatten
- set_max_area
- set_minimize_tree_delay
- set_resource_allocation
- set_structure
- uniquify
- compile_auto_ungroup_area_num_cells
- compile_auto_ungroup_override_wlm
- compile_implementation_selection
- compile_top_all_paths

Feedback

b

bound_attributes

Contains attributes related to bound.

Description

Contains attributes related to bound.

You can use *get_attribute* to determine value of an attribute, and use *report_attribute* to get a report of all attributes on specified object. Specified with *list_attribute -class bound -application*, the definition of attributes can be listed.

Bound Attributes

Specifies the *width*:*height* ratio of a bound.

The data type of *aspect_ratio* is double.

This attribute is read-only.

Specifies the bounding-box of a bound. The *bbox* is represented by a *rectangle*.

The format of a *rectangle* specification is {{llx lly} {urx ury}}, which specifies the lower-left and upper-right corners of the rectangle.

The data type of *bbox* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies the lower-left corner of the bounding-box of a bound.

The *bbox_ll* is represented by a *point*. The format of a *point* specification is {x y}.

You can get the *bbox_ll* of a bound, by accessing the first element of its *bbox*.

The data type of *bbox_ll* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies x coordinate of the lower-left corner of the bounding-box of a bound.

The data type of *bbox_llx* is double.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies y coordinate of the lower-left corner of the bounding-box of a bound.

Feedback

The data type of *bbox_lly* is double.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies the upper-right corner of the bounding-box of a bound.

The *bbox_ur* is represented by a *point*. The format of a *point* specification is {x y}.

You can get the *bbox_ur* of a bound, by accessing the second element of its *bbox*.

The data type of *bbox_ur* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies x coordinate of the upper-right corner of the bounding-box of a bound.

The data type of *bbox_urx* is double.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies y coordinate of the upper-right corner of the bounding-box of a bound.

The data type of *bbox_ury* is double.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies color to draw a move bound and its associated instances.

The data type of *color* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies dimension of a group bound.

Its format is {*width height*}.

The data type of *dimension* is string.

This attribute is read-only.

Specifies effort to bring cells closer inside an auto group bound.

Its valid values can be:

- **`low`**
- **`medium`**

- **high**
- **ultra**

The data type of *effort* is string.

This attribute is read-only.

Specifies whether the exclusive movebound is a multi-height row region.

The data type of *is_mhr* is boolean.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies name of a bound object.

The data type of *name* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies number of hard macro cells inside a bound.

The data type of *number_of_hard_macro* is integer.

This attribute is read-only.

Specifies number of standard cells inside a bound.

The data type of *number_of_standard_cell* is integer.

This attribute is read-only.

Specifies object class name of a bound, which is *bound*.

The data type of *object_class* is string.

This attribute is read-only.

Specifies object type of a bound, which can be *move_bound*, *auto_group_bound*, or *group_bound*.

The data type of *object_type* is string.

This attribute is read-only.

Specifies point list of a move bound's boundary.

The data type of *points* is string.

Feedback

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies type of a move bound or a group bound.

The data type of *type* is string.

Its valid values are:

- **soft**
- **hard**
- **exclusive**

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies the ratio of total area size of associated instances to the area of a move bound.

The data type of *utilization* is double.

This attribute is read-only.

See Also

- get_attribute
- list_attributes
- report_attribute
- set_attribute

C

cell_attributes

Contains attributes that can be placed on a cell.

Description

Contains attributes that can be placed on a cell.

There are a number of commands used to set attributes, however, most attributes can be set with the *set_attribute* command. If the attribute definition specifies a *set* command, use

it to set the attribute. Otherwise, use *set_attribute*. If an attribute is read-only, you cannot set it.

To determine the value of an attribute, use the *get_attribute* command. To remove attributes, use the *remove_attribute* command.

For a more detailed explanation of an attribute, refer to the manual pages of the appropriate *set* command. For information on all attributes, refer to the *attributes* manual page.

Cell Attributes

async_set_reset_q

Establishes the value (0 or 1) that should be assigned to the q output of an inferred register if set and reset are both active at the same time. To be used with *async_set_reset_qn*. Use these attributes only if you have used the *one_hot* or *one_cold* attributes/directives in your HDL description *and* your technology library is written using pre-V3.0a syntax; *or* if your technology library does not use a consistent convention for q and qn when set and reset are both active. If a V3.0a or later syntax technology library is used, then by default if set and reset are both active at the same time Design Compiler will use the convention of the selected technology library (*target_library*). Set with *set_attribute*.

Note: If you are unsure whether or not your technology library uses V3.0a syntax, ask your ASIC vendor.

async_set_reset_qn

Establishes the value (0 or 1) that should be assigned to the qn output of an inferred register if set and reset are both active at the same time. To be used with *async_set_reset_q*. Use these attributes only if you have used the *one_hot* or *one_cold* attributes/directives in your HDL description *and* your technology library is written using pre-V3.0a syntax; *or* if your technology library does not use a consistent convention for q and qn when set and reset are both active. If a V3.0a or later syntax technology library is used, then by default if set and reset are both active at the same time Design Compiler will use the convention of the selected technology library (*target_library*). Set with *set_attribute*.

Note: If you are unsure whether or not your technology library uses V3.0a syntax, ask your ASIC vendor.

disable_timing *

Disables the timing arcs of a cell. This has the same effect on timing as not having the arc in the library. Set with the *set_disable_timing* command.

dont_touch

Identifies cells to be excluded from optimization. Values are *true* (the default) or *false*. Cells with the *dont_touch* attribute set to *true* are not modified or replaced during *compile*. Setting *dont_touch* on a hierarchical cell sets the attribute on all cells below it. Set with *set_dont_touch*.

flip_flop_type

Stores the name of the specified flip-flop to be converted from the *target_library*. The *compile* command automatically converts all tagged flip-flops to the specified (or one similar) type. Set with *set_register_type -flip_flop flip_flop_name [cell_list]*.

flip_flop_type_exact

Stores the name of the specified flip-flop to be converted from the *target_library*. The *compile* command automatically converts all tagged flip-flops to the exact flip-flop type. Set with *set_register_type -exact -flip_flop flip_flop_name [cell_list]*.

is_black_box

Set to *true* if the cell's reference is not linked to a design.

This attribute is read-only and cannot be modified.

is_combinational

Set to *true* if all cells of a design and all designs in its hierarchy are combinational. A cell is combinational if it is non-sequential or non-tristate and all of its outputs compute a combinational logic function. The *report_lib* command will report such a cell as not a black-box.

This attribute is read-only and cannot be modified.

is_dw_subblock

Set to *true* if the object (a cell, a reference, or a design) is a DW subblock that was automatically elaborated.

This attribute is read-only and cannot be modified.

Note: DW subblocks that are manually elaborated will not have this attribute.

is_hierarchical

Set to *true* if the design contains leaf cells or other levels of hierarchy.

This attribute is read-only and cannot be modified.

Feedback

is_mapped

Set to *true* if the cell is not generic logic.

This attribute is read-only and cannot be modified.

is_sequential

Set to *true* if the cell is sequential. A cell is sequential if it is not combinational.

This attribute is read-only and cannot be modified.

is_synlib_module

Set to *true* if the object (a cell, a reference, or a design) refers to an unmapped module reference or if the object is (or refers to) a design that was automatically elaborated from a synlib module or a synlib operator.

This attribute is read-only and cannot be modified.

Note: synlib modules that are manually elaborated will not have this attribute.

is_synlib_operator

Set to *true* if the object (a cell or a reference) is a synthetic library operator reference.

This attribute is read-only and cannot be modified.

is_test_circuitry

Set by *insert_dft* on the scan cells and nets added to a design during the addition of test circuitry.

This attribute is read-only and cannot be modified.

is_unmapped

true if the cell is generic logic.

This attribute is read-only and cannot be modified.

latch_type_exact

Stores the name of the specified latch to be converted from the *target_library*. The *compile* command automatically converts all tagged latches to the exact latch type. Set with *set_sequential_type -latch latch_name [cell_list].*

map_only

When set to *true*, *compile* will attempt to map the object exactly in the target library, and will exclude the object from logic-level optimization (flattening and structuring). The default is *false*. Set with *set_map_only*.

mask_layout_type

Specifies the mask layout type of a cell.

This attribute is read-only and cannot be modified.

max_fall_delay

A floating point value that specifies the maximum falling delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_max_delay*.

max_rise_delay

A floating point value that specifies the maximum rising delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_max_delay*.

max_time_borrow

A floating point number that establishes an upper limit for time borrowing; that is, it prevents the use of the entire pulse width for level-sensitive latches. Units are those used in the technology library. Set with *set_max_time_borrow*.

min_fall_delay

A floating point value that specifies the minimum falling delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_min_delay*.

min_rise_delay

A floating point value that specifies the minimum rising delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_min_delay*.

ref_name

The reference name of a cell.

This attribute is read-only and cannot be modified.

scan

When *true*, specifies that the cell is always replaced by an equivalent scan cell during *insert_dft*. When *false*, the cell is not replaced. Set with *set_scan*.

scan_chain

Includes the specified cells of the referenced design in the scan-chain whose index is the value of this attribute. Set with *set_scan_chain*.

test_dont_fault

Specifies cells not faulted during test pattern generation. If no command options are specified, this attribute is set for both "stuck-at-0" and "stuck-at-1" faults. Set with *set_test_dont_fault*.

Feedback

test_isolate

Indicates that the specified sequential cells, pins, or ports are to be logically isolated and considered untestable during test design rule checking by *check_test*. When this attribute is set on a cell, it is also placed on all pins of that cell. Do not set this attribute on a hierarchical cell. Use *report_test -assertions* for a report on isolated objects. Set with *set_test_isolate*.

Note: Setting this attribute suppresses the warning messages associated with the isolated objects.

test_routing_position

Specifies the preferred routing order of the scan-test signals of the identified cells. Set with *set_test_routing_order*.

ungroup

Removes a level of hierarchy by exploding the contents of the specified cell in the current design. If specified on a reference object, cells using that reference are ungrouped during *compile*. Set with *set_ungroup*.

See Also

- get_attribute
- remove_attribute
- set_attribute
- attributes

---

clock_attributes

Contains attributes placed on clocks.

Description

Contains attributes that can be placed on clocks.

To set an attribute, use the command identified in the individual description of that attribute. To determine the value of an attribute, use the *get_attribute* command. To remove attributes, use the *remove_attribute* command.

For a more detailed explanation of an attribute, refer to the manual pages of the appropriate *set* command. For information on all attributes, refer to the *attributes* manual page.

Clock Attributes

dont_touch_network

When a design is optimized, *compile* assigns *dont_touch* attributes to all cells and nets in the transitive fanout of *dont_touch_network* ports. The *dont_touch* assignment stops at the boundary of storage elements. An element is recognized as storage only if it has setup or hold constraints. Set with *set_dont_touch_network*.

fix_hold

Specifies that *compile* should attempt to fix hold violations for timing endpoints related to this clock. Set with *set_fix_hold*.

max_fall_delay

A floating point value that specifies the maximum falling delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_max_delay*.

max_rise_delay

A floating point value that specifies the maximum rising delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_max_delay*.

max_time_borrow

A floating point number that establishes an upper limit for time borrowing; that is, it prevents the use of the entire pulse width for level-sensitive latches. Units are those used in the technology library. Set with *set_max_time_borrow*.

min_fall_delay

A floating point value that specifies the minimum falling delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_min_delay*.

min_rise_delay

A floating point value that specifies the minimum rising delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_min_delay*.

period

Assigns a value to the clock period. The clock period (or cycle time) is the shortest time during which the clock waveform repeats. For a simple waveform with one rising and one falling edge, the period is the difference between successive rising edges. Set with *create_clock -period period_value*.

hold_uncertainty

Specifies a negative uncertainty from the edges of the ideal clock waveform. Set with *set_clock_uncertainty -hold*.

Feedback

setup_uncertainty

Specifies a positive uncertainty from the edges of the ideal clock waveform. Set with *set_clock_uncertainty -setup*.

propagated_clock

Specifies that the clock edge times be delayed by propagating the values through the clock network. If this attribute is not present, ideal clocking is assumed. Set with *set_propagated_clock*.

See Also

- get_attribute
- remove_attribute
- attributes

---

core_area_attributes

Contains attributes related to core area.

Description

Contains attributes related to core area.

You can use *get_attribute* to determine value of an attribute, and use *report_attribute* to get a report of all attributes on specified object. Specified with *list_attribute -class core_area -application*, the definition of attributes can be listed.

Core Area Attributes

Specifies area of a core area object.

The data type of *area* is double.

This attribute is read-only.

Specifies the bounding-box of a core area. The *bbox* is represented by a *rectangle*.

The format of a *rectangle* specification is {{llx lly} {urx ury}}, which specifies the lower-left and upper-right corners of the rectangle.

The data type of *bbox* is string.

This attribute is read-only.

Specifies point list of a core area's boundary.

The data type of *boundary* is string.

This attribute is read-only.

Specifies Milkyway design ID in which a core area object is located.

The data type of *cell_id* is integer.

This attribute is read-only.

Specifies the direction to orient the rows in a core area.

The valid values can be horizontal and vertical.

The data type of *direction* is string.

This attribute is read-only.

Specifies whether a core area contains pairs of rows with one row flipped in each pair.

The data type of *is_double_back* is boolean.

This attribute is read-only.

Specifies whether to flip the first row at the bottom of a horizontal core area or at the left of a vertical core area.

The data type of *is_flip_first_row* is boolean.

This attribute is read-only.

Specifies whether the pairing of rows starts at the bottom of a horizontal core area or at the left of a vertical core area.

The data type of *is_start_first_row* is boolean.

This attribute is read-only.

Specifies name of a core area object.

The data type of *name* is string.

This attribute is read-only.

Specifies number of rows inside a core area.

The data type of *num_rows* is integer.

This attribute is read-only.

Specifies object class name of a core area, which is *core_area*.

The data type of *object_class* is string.

This attribute is read-only.

Specifies object ID in Milkyway design file.

Feedback

The data type of *object_id* is integer.

This attribute is read-only.

Specifies the ratio of total height (if it's a horizontal core area) or width (otherwise) of rows to the height or width of a core area.

The data type of *row_density* is string.

This attribute is read-only.

Specifies height of tile cell used in a core area.

The data type of *tile_height* is double.

This attribute is read-only.

Specifies width of tile cell used in a core area.

The data type of *tile_width* is double.

This attribute is read-only.

See Also

- get_attribute
- list_attributes
- report_attribute
- set_attribute

d

design_attributes

Description

Contains attributes that can be placed on a design.

There are a number of commands used to set attributes. Most attributes, however, can be set with the *set_attribute* command. If the attribute definition specifies a *set* command, use it to set the attribute. Otherwise, use *set_attribute*. If an attribute is "read only," it cannot be set by the user.

To determine the value of an attribute, use the *get_attribute* command. To remove attributes, use the *remove_attribute* command.

For a more detailed explanation of an attribute, refer to the manual pages of the appropriate *set* command. For information on all attributes, refer to the *attributes* manual page.

Design Attributes

async_set_reset_q

Establishes the value (0 or 1) that should be assigned to the q output of an inferred register if set and reset are both active at the same time. To be used with *async_set_reset_qn*. Use these attributes only if you have used the *one_hot* or *one_cold* attributes/directives in your HDL description *and* your technology library is written using pre-V3.0a syntax; *or* if your technology library does not use a consistent convention for q and qn when set and reset are both active. If a V3.0a or later syntax technology library is used, then, by default, if set and reset are both active at the same time Design Compiler will use the convention of the selected technology library (*target_library*). Set with *set_attribute*.

Note: If you are unsure whether your technology library uses V3.0a syntax, ask your ASIC vendor.

async_set_reset_qn

Establishes the value (0 or 1) that should be assigned to the qn output of an inferred register if set and reset are both active at the same time. To be used with *async_set_reset_q*. Use these attributes only if you have used the *one_hot* or *one_cold* attributes/directives in your HDL description *and* your technology library is written using pre-V3.0a syntax; *or* if your technology library does not use a consistent convention for q and qn when set and reset are both active. If a V3.0a or later syntax technology library is used, then, by default, if set and reset are both active at the same time Design Compiler will use the convention of the selected technology library (*target_library*). Set with *set_attribute*.

Note: If you are unsure whether your technology library uses V3.0a syntax, ask your ASIC vendor.

balance_registers

Determines whether the registers in a design are retimed during *compile*. When *true* (the default value), *compile* invokes the *balance_registers* command, which moves registers to minimize the maximum register-to-register delay. Set this attribute to *false*, or remove it, to disable this behavior.

Set with *set_balance_registers*.

If your design contains generic logic, you should ensure that all components are mapped to cells from the library before setting the *balance_registers* attribute.

boundary_optimization

Enables *compile* to optimize across hierarchical boundaries. Hierarchy is ignored during optimization for designs with this attribute set to *true*. Set with *set_boundary_optimization*.

default_flip_flop_type

Specifies the default flip-flop type for the current design. During the mapping process, *compile* tries to convert all unmapped flip-flops to this type. If *compile* is unable to use this flip-flop, it maps these cells into the smallest flip-flop possible. Set with *set_register_type -flip_flop flip_flop_name*.

default_flip_flop_type_exact

Specifies the exact default flip-flop type for the current design. During the mapping process, *compile* converts unmapped flip-flops to the exact flip-flop type specified here. Set with *set_register_type -exact -flip_flop flip_flop_name*.

default_latch_type_exact

Specifies the exact default latch type for the current design. During the mapping process, *compile* converts unmapped latches to the exact latch type specified here. Set with *set_register_type -exact -latch latch_name*.

design_type

Specifies the type of the current design. Allowed values are *fsm* (finite state machine); *pla* (programmable logic array); *equation* (Boolean logic); or *netlist* (gates). This attribute is read-only and cannot be set by the user.

dont_touch

Identifies designs that are to be excluded from optimization. Values are *true* (the default) or *false*. Designs with the *dont_touch* attribute set to *true* are not modified or replaced during *compile*. Setting *dont_touch* on a design has an effect only when the design is instantiated within another design as a level of hierarchy; setting *dont_touch* on the top-level design has no effect. Set with *set_dont_touch*.

flatten

When set to *true*, determines that a design is to be flattened during *compile*. By default, a design is not flattened. Set with *set_flatten*.

flatten_effort

Defines the level of CPU effort that *compile* uses to flatten a design. Allowed values are *low* (the default), *medium*, or *high*. Set with *set_flatten*.

flatten_minimize

Defines the minimization strategy used for logic equations. Allowed values are *single_output*, *multiple_output*, or *none*. Set with *set_flatten*.

flatten_phase

When *true*, allows logic flattening to invert the phase of outputs during *compile*. By default, logic flattening does not invert the phase of outputs. Used only if the *flatten* attribute is set. Set with *set_flatten*.

hdl_template

Specifies the original RTL design name for a parameterized design after running the *analyze* and *elaborate* commands. This attribute is not available after running the *uniquify* command on the design. After running the *uniquify* command, use the *original_design_name* attribute to get the original design name.

is_combinational

true if all cells of a design and all designs in its hierarchy are combinational. A cell is combinational if it is non-sequential or non-tristate and all of its outputs compute a combinational logic function. The *report_lib* command will report such a cell as not a black-box. This attribute is read-only and cannot be set by the user.

is_dw_subblock

true if the object (a cell, a reference, or a design) is a DW subblock that was automatically elaborated. This attribute is read-only and cannot be set by the user.

NOTE: DW subblocks that are manually elaborated will not have this attribute.

is_hierarchical

true if the design contains leaf cells or other levels of hierarchy. This attribute is read only and cannot be set by the user.

is_mapped

true if all the non-hierarchical cells of a design are mapped to cells in a technology library. This attribute is read-only and cannot be set by the user.

is_sequential

true if any cells of a design or designs in its hierarchy are sequential. A cell is sequential if it is not combinational (if any of its outputs depend on previous inputs). This attribute is read-only and cannot be set by the user.

is_synlib_module

true if the object (a cell, a reference, or a design) refers to an unmapped module reference or if the object is (or refers to) a design that was automatically elaborated from a synlib module or a synlib operator. This attribute is read-only and cannot be set by the user.

NOTE: synlib modules that are manually elaborated will not have this attribute.

is_unmapped

true if any of the cells are not linked to a design or mapped to a technology library. This attribute is read-only and cannot be set by the user.

local_link_library

A string that contains a list of design files and libraries to be added to the beginning of the *link_library* whenever a *link* operation is performed. Set with *set_local_link_library*.

map_only

When set to *true*, *compile* will attempt to map the object exactly in the target library, and will exclude the object from logic-level optimization (flattening and structuring). The default is *false*. Set with *set_map_only*.

max_capacitance

A floating point number that sets the maximum capacitance value for input, output, or bidirectional ports; or for designs. The units must be consistent with those of the technology library used during optimization. Set with *set_max_capacitance*.

max_total_power

A floating point number that specifies the maximum target total power for the *current_design*. Total power is defined as the sum of dynamic and leakage power. If this attribute is specified more than once for a design, the latest value is used. Set with *set_max_total_power*.

minimize_tree_delay

When *true* (the default value), *compile* restructures expression trees in the *current_design* or in a list of specified designs, to minimize tree delay. Set this attribute to *false* for any designs that you do not wish to be restructured. Set with *set_minimize_tree_delay*.

model_map_effort

Specifies the relative amount of CPU time to be used by *compile* during modeling, typically for synthetic library implementations. Values are *low*,

medium, and *high*, or 1, 2, and 3. If *model_map_effort* is not set, the value of *synlib_model_map_effort* is used. Set with *set_model_map_effort*.

model_scale

A floating point number that sets the model scale factor for the *current_design*. Set with *set_model_scale*.

optimize_registers

When *true* (the default value), *compile* automatically invokes the Behavioral Compiler *optimize_registers* command to retime the design during optimization. Setting the attribute to *false* disables this behavior.

Your design cannot contain generic logic at the instant *optimize_registers* is invoked during *compile*. Set with *set_optimize_registers*.

original_design_name

Specifies the original design name for a parameterized design after running the *uniquify* command.

part

A string value that specifies the Xilinx part type for a design. For valid part types, refer to the Xilinx *XC4000 Databook*. Set with *set_attribute*.

resource_allocation

Indicates the type of resource allocation to be used by *compile* for the *current_design*. Allowed values are *none*, indicating no resource sharing; *area_only*, indicating resource sharing with tree balancing without considering timing constraints; *area_no_tree_balancing*, indicating resource sharing without tree balancing and without considering timing constraints; and *constraint_driven* (the default), indicating resource sharing so that timing constraints are met or not worsened. The value of this attribute overrides the value of the variable *hlo_resource_allocation* for the *current_design*. Set with *set_resource_allocation*.

resource_implementation

Indicates the type of resource implementation to be used by *compile* for the *current_design*. Allowed values are *area_only*, indicating resource implementation without considering timing constraints; *constraint_driven*, indicating resource implementation so that timing constraints are met or not worsened; and *use_fastest*, indicating resource implementation using the fastest implementation initially and using components with smaller area later in uncritical parts of the design. The value of this attribute overrides the value of the variable *hlo_resource_implementation* for the *current_design*. Set with *set_resource_implementation*.

structure

Determines if a design is to be structured during *compile*. If *true*, adds logic structure to a design by adding intermediate variables that are factored out of the design's equations. Set with *set_structure*.

structure_boolean

Enables the use of Boolean (non-algebraic) techniques during the structuring phase of optimization. This attribute is ignored if the *structure* attribute is *false*. Set with *set_structure*.

structure_timing

Enables timing constraints to be considered during the structuring phase of optimization. This attribute is ignored if the *structure* attribute is *false*. Set with *set_structure*.

ungroup

Removes a level of hierarchy from the current design by exploding the contents of the specified cell in the current design. Set with *set_ungroup*.

wired_logic_disable

When *true*, disables the creation of wired OR logic during *compile*. The default is *false*; if this attribute is not set, wired OR logic will be created if appropriate. Set with *set_wired_logic_disable*.

wire_load_model_mode

Determines which wire load model to use to compute wire capacitance, resistance, and area for nets in a hierarchical design that has different wire load models at different hierarchical levels. Allowed values are *top*, use the wire load model at the top hierarchical level; *enclosed*, use the wire load model on the smallest design that encloses a net completely; and *segmented*, break the net into segments, one within each hierarchical level. In the *segmented* mode, each net segment is estimated using the wire load model on the design that encloses that segment. The *segmented* mode is not supported for wire load models on clusters. If a value is not specified for this attribute, *compile* searches for a default in the first library in the link path. If none is found, *top* is the default. Set with *set_wire_load*.

See Also

- get_attribute
- remove_attribute

Feedback

- set_attribute
- attributes

---

die_area_attributes

Contains attributes related to die area.

Description

Contains attributes related to die area.

You can use *get_attribute* to determine value of an attribute, and use *report_attribute* to get a report of all attributes on specified object. Specified with *list_attribute -class die_area -application*, the definition of attributes can be listed.

Die Area Attributes

Specifies the bounding-box of a die area. The *bbox* is represented by a *rectangle*.

The format of a *rectangle* specification is {{llx lly} {urx ury}}, which specifies the lower-left and upper-right corners of the rectangle.

The data type of *bbox* is string.

This attribute is read-only.

Specifies point list of a die area's boundary.

The data type of *boundary* is string.

This attribute is read-only.

Specifies Milkyway design ID in which a die area object is located.

The data type of *cell_id* is integer.

This attribute is read-only.

Specifies name of a die area object.

The data type of *name* is string.

This attribute is read-only.

Specifies object class name of a die area, which is *die_area*.

The data type of *object_class* is string.

This attribute is read-only.

Specifies object ID in Milkyway design file.

Feedback

The data type of *object_id* is integer.

This attribute is read-only.

See Also

- get_attribute
- list_attributes
- report_attribute
- set_attribute

f

floorplan_data_attributes

Contains attributes related to floorplan data on different modules in separate views.

Description

Contains attributes related to floorplan data on different modules in separate views.

Floorplan data attributes are read-only. You can use *get_floorplan_data* to determine value of an attribute, and use *report_floorplan_data* to get a report of all the floorplan data on module objects in a specified view. If you want to know the list of valid floorplan data names, you can use *list_floorplan_data*.

Floorplan Data Attributes

area

Specifies the area of a module. This attribute does not exist on a logical module.

The tool calculates the attribute based on cell boundary.

aspect_ratio

Specifies the *height*:*width* ratio of a module. This attribute does not exist on a logical module.

hard_macro_area_percentage

Specifies ratio of sum of *area* of hard macros in a module to top module's *physical_area*.

height

Specifies the height of a module. This attribute only exists on soft macros or hard macros in *logical* view.

The tool calculates the attribute based on the *bbox* of a module.

macro_area_percentage

Specifies the ratio of sum of *area* of macros in a module to top module's *physical_area*.

Note that the definition of *macro_area_percentage* in this document is different from that in *cell_attributes.3*. Compared to the definition in *cell_attributes.3*, the numerator of ratio remains while the denominator is *physical_area* of top module instead of *physical_area* of the specified module.

number_of_black_box

Specifies the count of black boxes in a module. This attribute does not exist on plan_group.

Whether a cell is a black box can be determined by the attribute *is_black_box*.

number_of_hard_macro

Specifies the count of hard macros in a module.

Whether a cell is a hard macro can be determined by the attribute *is_hard_macro*.

number_of_io_cell

Specifies the count of io cells in a module. This attribute does not exist on *plan_group*.

The tool counts cells in when its *mask_layout_type* is *io_pad*, *corner_pad*, *pad_filler*, or *flip_chip_pad*.

number_of_macro

Specifies the count of macros in a module. This attribute does not exist on *plan_group*.

The tool counts cells in when its *mask_layout_type* is *macro*.

number_of_standard_cell

Specifies the count of standard cells in a module.

The tool counts cells in when its *mask_layout_type* matches **std**.

physical_area

Specifies the physical area of a module.

The physical area of a hard macro is equal to its *area*.

In *one_level* or *logical* view, the physical area of a soft macro is equal to its *area*. However, in *physical* view, physical area is the sum of *physical_area* of macros and *area* of standard cells inside the specified soft macro.

The physical area of a plan group is sum of *area* of macros and standard cells in it.

For a top module or a logical module, rules for calculating *physical_area* are applied as:

In a *one_level* view, only children in the current level are counted and children should be either macro or standard cell. The physical area of a module in *one_level* is sum of the *area* of the counted children.

In a *logical* view, the hierarchy tree in the module is traversed and the *area* of macros and standard cells in the tree will be added.

In the *physical* view, the hierarchy tree in the module is traversed and the *area* of standard cells and *physical_area* of macros in the tree will be added.

physical_area_percentage_in_top_design

Specifies the ratio of *physical_area* of a module to top module's *physical_area*.

utilization

Specifies the utilization of a module.

The tool calculates the attribute using *physical_area*:*area* ratio of a module. This attribute does not exist on a logical module or a top module.

width

Specifies the width of a module. This attribute only exists on soft macros or hard macros in *logical* view.

The tool calculates the attribute based on the *bbox* of a module.

l

layer_attributes

Contains attributes related to layer.

Feedback

Description

Contains attributes related to layer.

You can use *get_attribute* to determine value of an attribute, and use *report_attribute* to get a report of all attributes on specified object. Specified with *list_attribute -class layer -application*, the definition of attributes can be listed.

Layer Attributes

Specifies detail information of data types on a layer.

The data type of *data_type_details* is string.

This attribute is read-only.

Specifies data types on a layer.

The data type of *data_types* is string.

This attribute is read-only.

Specifies the default width of any dimension of an object on a layer.

The data type of *defaultWidth* is float.

This attribute is read-only.

Specifies the threshold for using a fat wire contact on a layer instead of the default contact.

The data type of *fatContactThreshold* is float.

This attribute is read-only.

Specifies the minimum distance required between wires on a layer when the widths of both wires are greater than or equal to *fatWireThreshold*.

The data type of *fatFatMinSpacing* is float.

This attribute is read-only.

Specifies the minimum distance required between wires on a layer when the width of one of the wires is greater than or equal to *fatWireThreshold*.

The data type of *fatThinMinSpacing* is float.

This attribute is read-only.

Specifies the threshold for using the fat wire spacing rule instead of the default spacing rule on a layer.

The data type of *fatWireThreshold* is float.

This attribute is read-only.

Feedback

Specifies the layer used for routing when there are multiple layers with the same *mask_name* value.

The data type of *isDefaultLayer* is integer.

This attribute is read-only.

Specifies whether layer is a routing layer.

The data type of *is_routing_layer* is boolean.

This attribute is read-only.

Defines the number that identifies a layer.

The data type of *layerNumber* is integer.

This attribute is read-only.

Defines the number that identifies a layer.

The data type of *layer_number* is integer.

This attribute is read-only.

Specifies type of a layer.

The data type of *layer_type* is string.

This attribute is read-only.

Specifies the physical layer associated with the specified layer object.

The data type of *mask_name* is string.

This attribute is read-only.

Specifies the floating-point number representing in amperes per centimeter the maximum current density a layer can carry.

The data type of *maxCurrDensity* is double.

This attribute is read-only.

Defines the maximum number of vias that can stack at the same point.

The data type of *maxStackLevel* is integer.

This attribute is read-only.

Specifies the minimum area rule of any dimension of an object on a layer.

The data type of *minArea* is float.

This attribute is read-only.

Feedback

Specifies the minimum separation distance between the edges of objects on a layer, if the objects are on different nets.

The data type of *minSpacing* is float.

This attribute is read-only.

Specifies the minimum width of any dimension of an object on a layer.

The data type of *minWidth* is float.

This attribute is read-only.

Specifies name of a layer object.

The data type of *name* is string.

This attribute is read-only.

Specifies object class name of a layer, which is *layer*.

The data type of *object_class* is string.

This attribute is read-only.

Specifies the predominant separation distance between the centers of objects on a layer.

The data type of *pitch* is float.

This attribute is read-only.

Specifies the preferred routing direction for a layer.

The data type of *preferred_direction* is string.

This attribute is read-only.

Specifies the maximum capacitance of a layer.

The data type of *unitMaxCapacitance* is double.

This attribute is read-only.

Specifies the maximum distance of a layer.

The data type of *unitMaxHeightFromSub* is double.

This attribute is read-only.

Specifies the maximum resistance of a layer.

The data type of *unitMaxResistance* is double.

This attribute is read-only.

Feedback

Specifies the maximum sidewall capacitance of a layer.

The data type of *unitMaxSideWallCap* is double.

This attribute is read-only.

Specifies the maximum thickness of a layer.

The data type of *unitMaxThickness* is double.

This attribute is read-only.

Specifies the minimum capacitance of a layer.

The data type of *unitMinCapacitance* is double.

This attribute is read-only.

Specifies the minimum distance of a layer.

The data type of *unitMinHeightFromSub* is double.

This attribute is read-only.

Specifies the minimum resistance of a layer.

The data type of *unitMinResistance* is double.

This attribute is read-only.

Specifies the minimum sidewall capacitance of a layer.

The data type of *unitMinSideWallCap* is double.

This attribute is read-only.

Specifies the minimum thickness of a layer.

The data type of *unitMinThickness* is double.

This attribute is read-only.

Specifies the nominal capacitance of a layer.

The data type of *unitNomCapacitance* is double.

This attribute is read-only.

Specifies the nominal distance of a layer.

The data type of *unitNomHeightFromSub* is double.

This attribute is read-only.

Specifies the nominal resistance of layer.

Feedback

The data type of *unitNomResistance* is double.

This attribute is read-only.

Specifies the nominal sidewall capacitance of a layer.

The data type of *unitNomSideWallCap* is double.

This attribute is read-only.

Specifies the nominal thickness of a layer.

The data type of *unitNomThickness* is double.

This attribute is read-only.

Specifies a layer's visibility.

The data type of *visible* is integer.

This attribute is read-only.

See Also

- get_attribute
- list_attributes
- report_attribute
- set_attribute

---

library_attributes

Contains attributes placed on libraries.

Description

Contains attributes that can be placed on a library.

To set library attributes, use the *set_attribute* command. To determine the value of an attribute, use the *get_attribute* command. To remove attributes, use the *remove_attribute* command.

For more details on library attributes, see the *Library Compiler Reference Manual*. For information on all attributes, refer to the *attributes* manual page.

Library Attributes

default_fanout_load

Fanout load of input pins in a library.

Feedback

default_inout_pin_cap

Capacitance of inout pins.

default_inout_pin_fall_res

Fall resistance of inout pins.

default_inout_pin_rise_res

Rise resistance of inout pins.

default_input_pin_cap

Capacitance of input pins.

default_intrinsic_fall

Intrinsic fall delay of timing arcs.

default_intrinsic_rise

Intrinsic rise delay of a timing arc.

default_max_fanout

Maximum fanout of pins.

default_max_transition

Maximum transition of pins.

default_min_porosity

Minimum porosity of designs.

default_output_pin_cap

Capacitance of output pins.

default_output_pin_fall_res

Fall resistance of output pins.

default_output_pin_rise_res

Rise resistance of output pins.

default_slope_fall

Fall sensitivity factor of a timing arc.

default_slope_rise

Rise sensitivity factor of a timing arc.

Feedback

k_process_drive_fall

Process scale factor applied to the fall resistance of timing arcs.

k_process_drive_rise

Process scale factor applied to the rise resistance of timing arcs.

k_process_intrinsic_fall

Process scale factor applied to the intrinsic fall delay of timing arcs.

k_process_intrinsic_rise

Process scale factor applied to the intrinsic rise delay of timing arcs.

k_process_pin_cap

Process scale factor applied to pin capacitance of timing arcs.

k_process_slope_fall

Process scale factor applied to the fall slope sensitivity of timing arcs.

k_process_slope_rise

Process scale factor applied to the rise slope sensitivity of timing arcs.

k_process_wire_cap

Process scale factor applied to the wire capacitance of timing arcs.

k_process_wire_res

Process scale factor applied to the wire resistance of timing arcs.

k_temp_drive_fall

Scale factor applied to timing arc fall resistance due to temperature variation.

k_temp_drive_rise

Scale factor applied to timing arc rise resistance due to temperature variation.

k_temp_intrinsic_fall

Scale factor applied to the intrinsic fall delay of a timing arc due to temperature variation.

k_temp_intrinsic_rise

Scale factor applied to the intrinsic rise delay of a timing arc due to temperature variation.

k_temp_pin_cap

Scale factor applied to pin capacitance due to temperature variation.

Feedback

k_temp_slope_fall

Scale factor applied to timing arc fall slope sensitivity due to temperature variation.

k_temp_slope_rise

Scale factor applied to timing arc rise slope sensitivity due to temperature variation.

k_temp_wire_cap

Scale factor applied to wire capacitance due to temperature variation.

k_temp_wire_res

Scale factor applied to wire resistance due to temperature variation.

k_volt_drive_fall

Scale factor applied to timing arc fall resistance due to voltage variation.

k_volt_drive_rise

Scale factor applied to timing arc rise resistance due to voltage variation.

k_volt_intrinsic_fall

Scale factor applied to the intrinsic fall delay of a timing arc due to voltage variation.

k_volt_intrinsic_rise

Scale factor applied to the intrinsic rise delay of a timing arc due to voltage variation.

k_volt_pin_cap

Scale factor applied to pin capacitance due to voltage variation.

k_volt_slope_fall

Scale factor applied to timing arc fall slope sensitivity due to voltage variation.

k_volt_slope_rise

Scale factor applied to timing arc rise slope sensitivity due to voltage variation.

k_volt_wire_cap

Scale factor applied to wire capacitance due to voltage variation.

k_volt_wire_res

Scale factor applied to wire resistance due to voltage variation.

Feedback

nom_process

Nominal process value used for library characterization. Fixed at 1.0 for most technology libraries.

nom_temperature

Nominal ambient temperature used for library characterization. Usually 25 degrees Celsius. Multipliers use the nominal value to determine the change in temperature between nominal and operating conditions.

nom_voltage

Nominal source voltage value used in library element characterization. Typically 5 volts for a CMOS library. Multipliers use the nominal value to determine the change in voltage between nominal and operating conditions.

no_sequential_degenerates

When *true*, disables mapping to degenerated flip-flops or latches (that is, devices that have some input pins connected to 0 or to 1). The default for the attribute is *false\P or nonexistence, implying that degenerate devices will be allowed by default in the library. This attribute may be overridden on a component-by-component basis by using the attribute on library cells.*

sequential_bridging

When *true*, enables *Design Compiler* to take a multiplexed flip-flop and bridge (that is, connect) the output to the input to get a desired functionality. The default for the attribute is *false* or nonexistent, implying that bridging will be disabled by default in the library. Bridging is required for mapping in cases where there is no flip-flop with internal feedback in the target library but one is desired in the HDL. This attribute may be overridden on a component-by-component basis by using the attribute on library cells.

NOTE: Setting this attribute to *true* can result in an increase in run times and memory consumption for Design Compiler. The increased run times depend on the number of flip-flops in the target library or libraries which have the attribute set.

See Also

- get_attribute
- remove_attribute
- set_attribute
- attributes

Feedback

library_cell_attributes

Contains attributes that can be placed on a library cell.

Description

Contains attributes that can be placed on a library cell.

To set an attribute, use the command identified in the individual description of that attribute. To determine the value of an attribute, use the *get_attribute* command. To remove attributes, use the *remove_attribute* command. If an attribute is "read-only," you cannot set it.

For a more detailed explanation of an attribute, refer to the manual pages of the appropriate *set* command. For information on all attributes, refer to the *attributes* manual page.

Library Cell Attributes

dont_touch

Identifies library cells to be excluded from optimization. Values are *true* (the default) or *false*. Library cells with the *dont_touch* attribute set to *true* are not modified or replaced during *compile*. Setting *dont_touch* on a hierarchical cell sets the attribute on all cells below it. Set with *set_dont_touch*.

dont_use

Disables the specified library cells so that they are not added to a design during *compile*. Set with *set_dont_use*.

no_sequential_degenerates

When *true*, disables mapping to versions of this latch or flip flop that have some input pins connected to 0 or to 1. Set with *set_attribute*. This attribute may also be set on the library itself, and that value will apply as the default for all registers in the library which do not have the attribute set individually.

preferred

Specifies the preferred library gate to use during technology translation when there are other gates with the same function in the target library. Set with *set_prefer*.

scan

When *true*, specifies that the instances of the library cell are always replaced by equivalent scan cells during *insert_dft*. When *false*, instances are not replaced. Set with *set_scan*.

Feedback

sequential_bridging

When *true*, enables Design Compiler to take a multiplexed flip-flop and bridge (that is, connect) the output to the input to get a desired functionality. The default is *false*, so this attribute must be set in order to enable the functionality. Bridging is required for mapping in cases where there is no flip-flop with internal feedback in the target library but one is desired in the HDL. Set with *set_attribute*. This attribute may also be set on the library itself, and that value will apply as the default for all registers in the library which do not have the attribute set individually.

NOTE: Setting this attribute to *true* can result in an increase in run times and memory consumption for Design Compiler. The increased run times depend on the number of flip-flops in the target library or libraries for which this attribute has been set.

See Also

- get_attribute
- remove_attribute
- set_attribute
- attributes

n

net_attributes

Contains attributes that can be placed on a net.

Description

Contains attributes that can be placed on a net.

To set an attribute, use the command identified in the individual description of that attribute. If an attribute is "read-only," the user cannot set it.

To determine the value of an attribute, use the *get_attribute* command. To remove attributes, use the *remove_attribute* command.

For a more detailed explanation of an attribute, refer to the manual pages of the appropriate *set* command. For information on all attributes, refer to the *attributes* manual page.

Net Attributes

ba_net_resistance

A floating point number that specifies the back-annotated net resistance on a net. Set with *set_resistance*.

bus_name

If the net is bussed, returns the full hierarchical name of the net bus. This attribute is *read only* and cannot be set by the user.

dont_touch

Identifies nets to be excluded from optimization. Values are *true* (the default) or *false*. Nets with the *dont_touch* attribute set to *true* are not modified or replaced during *compile*. Set with *set_dont_touch*.

groute_length

Get the global routing length of the net if global route is done for the net.

is_bussed

Returns true if the net is part of a bussed net. This attribute is *read only* and cannot be set by the user.

is_test_circuitry

Set by *insert_dft* on the scan cells and nets added to a design during the addition of test circuitry. This attribute is read-only and cannot be set by the user.

load *

A floating point number that specifies the wire load value on a net. The total load on a net is the sum of all the loads on pins, ports, and wires associated with that net. This attribute represents only the wire load; pin and port loads are not included in the attribute value unless the attribute *subtract_pin_load* is set to *true* for the same net. Set with *set_load*.

route_mode

The rerouting mode of a net. The attribute could be *normal*, *minorchange*, or*freeze*. The attribute could be set by *set_net_routing_rule* command with *-reroute* option.

static_probability

A floating point number that specifies the percentage of time that the signal is in the logic 1 state; this information is used by *report_power*. If this attribute is not set, *report_power* will use the default value of 0.5, indicating that the signal is in the logic 1 state half the time. Set with *set_switching_activity*.

subtract_pin_load

Causes *compile* to reduce the wire load value of a net by an amount equal to its pin load. Specifies that the *load* attribute includes the capacitances of all pins on the net. If the resulting wire load is negative, it is set to zero. Set with *set_load -subtract_pin_load*.

route_guide_group_id

Specifies the route guide group id for a flat net. The data type of *route_guide_group_id* is int. The valid value range of *route_guide_group_id* is 1~8. This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

switching_activity

A positive floating point number that specifies the switching activity; that is, the number of zero-to-one and one-to-zero transitions within a library time unit period. This information is used by *report_power*; if this attribute is not set, *report_power* will use the default value of 2*(static_probability)(1 - static_probability). The default will be scaled by any associated clock signal (if one is available). Set with *set_switching_activity*.

wired_and

One of a set of two wired logic attributes that includes *wired_or*. When present and set to *true*, *wired_and* determines that the associated net has more than one driver and implements a wired AND function. Wired logic attributes cannot be manually set by the user. To cause wired logic attributes to be added to a netlist design that contains multiply-driven nets, you have two alternatives: 1. execute *compile* or *translate* on the design; or 2. specify the wired logic types using a resolution function in the HDL file.

wired_or

One of a set of two wired logic attributes that includes *wired_and*. When present and set to *true*, *wired_or* determines that the associated net has more than one driver and implements a wired OR function. Wired logic attributes cannot be manually set by the user. To cause wired logic attributes to be added to a netlist design that contains multiply-driven nets, you have two alternatives: 1. execute *compile* or *translate* on the design; or 2. specify the wired logic types using a resolution function in the HDL file.

See Also

- get_attribute
- remove_attribute
- attributes

Feedback

p

physical_bus_attributes

Contains attributes related to physical buses.

Description

Contains attributes related to physical buses.

You can use the *get_attribute* command to determine the value of an attribute and use the *report_attribute* command to get a report of all attributes on a specified physical bus. To list the definitions of the attributes, use the *list_attribute -class physical_bus -application* command.

Physical Bus Attributes

Specifies the Milkyway design ID in which a physical bus object is located.

This attribute is read-only.

Specifies the name of a physical bus object.

This attribute is read-only.

Specifies the number of nets in a physical bus object.

You can get the nets associated with a physical bus object by using the *get_nets -of_object <physical_bus_object>* command.

This attribute is read-only.

Specifies the object class name of a physical bus object, which is *physical_bus*.

This attribute is read-only.

Specifies the object ID in the Milkyway design file.

This attribute is read-only.

See Also

- get_attribute
- list_attributes
- report_attribute

Feedback

physical_lib_pin_attributes

Describes the attributes related to physical library pins.

Description

This man page describes the attributes related to physical library pins.

You can use the *get_attribute* command to determine value of an attribute, and use the *report_attribute* command to get a report of all attributes on a specified object. To see the attributes defined for physical library pins, use the *list_attribute -class physical_lib_pin -application* command.

All of these attributes are read-only.

Physical Library Pin Attributes

cell_id

Specifies Milkyway design ID in which a physical_lib_pin object is located.

The data type of the *cell_id* attribute is integer.

cell_name

Specifies name of the Milkyway design in which a physical_lib_pin object is located.

The data type of the *cell_name* attribute is string.

direction

Specifies the direction of a physical_lib_pin object.

The data type of the *direction* attribute is string.

full_name

Specifies the full name of a physical_lib_pin object. The full name of a physical_lib_pin object is composed of the library name (*lib_name* attribute), the cell name (*cell_name* attribute), and the pin name (*name* attribute).

The data type of the *full_name* attribute is string.

is_diode

Indicates whether the specified object is a diode.

The data type of the *is_diode* attribute is Boolean.

lib_name

Specifies the library name of a physical_lib_pin object.

The data type of the *lib_name* attribute is string.

lib_path

Specifies the UNIX path name of the library that contains the physical_lib_pin object.

The data type of the *lib_path* attribute is string.

name

Specifies the name of a physical_lib_pin object.

The data type of the *name* attribute is string.

object_class

Specifies the object class name of a physical_lib_pin, which is *physical_lib_pin*.

The data type of the *object_class* attribute is string.

object_id

Specifies the ID of the physical_lib_pin object in the Milkyway design database.

The data type of the *object_id* attribute is integer.

pin_type

Specifies the pin type of a physical_lib_pin object.

The data type of the *pin_type* attribute is string.

pg_pin_weight

A float attribute representing the PG pin weight. It should be between 0.1 and 25.0. And the fraction precision is 0.1, e.g. 0.11 will be cut off as 0.1.

See Also

- get_attribute
- report_attribute

---

pin_attributes

Contains attributes placed on pins.

Description

Contains attributes that can be placed on a pin.

There are a number of commands used to set attributes; however, most attributes can be set with the *set_attribute* command. If the attribute definition specifies a *set* command, use it to set the attribute. Otherwise, use the *set_attribute* command. If an attribute is *read only*, it cannot be set by the user.

To determine the value of an attribute, use the *get_attribute* command. To remove attributes, use the *remove_attribute* command.

For a detailed information of an attribute, see the man pages of the appropriate *set* command. For information on all attributes, see the *attributes* man page.

Pin Attributes

bus_name

If the pin is bussed, returns the full hierarchical name of the pin bus. This attribute is *read only* and cannot be set by the user.

clock

The clock attribute indicates whether an input pin is a clock pin.

```
clock : true | false
```

The *true* value specifies the pin as a clock pin. The *false* value specifies the pin as not a clock pin, even though it might have the clock characteristics.

The following example defines pin CLK2 as a clock pin.

```
pin(CLK2) {
  direction : input ;
  capacitance : 1.0 ;
  clock : true ;
}
```

disable_timing

Disables timing arcs. This has the same affect on timing as not having the arc in the library. Set with the *set_disable_timing* command.

is_bussed

Returns true if the pin is part of a bussed pin. This attribute is *read only* and cannot be set by the user.

is_diode

Tells if the pin is a diode pin.

max_fall_delay

A floating point value that specifies the maximum falling delay on ports, clocks, pins, cells, or on paths between such objects. Set with the *set_max_delay* command.

max_rise_delay

A floating point value that specifies the maximum rising delay on ports, clocks, pins, cells, or on paths between such objects. Set with the *set_max_delay* command.

min_fall_delay

A floating point value that specifies the minimum falling delay on ports, clocks, pins, cells, or on paths between such objects. Set with the *set_min_delay* command.

min_rise_delay

A floating point value that specifies the minimum rising delay on ports, clocks, pins, cells, or on paths between such objects. Set with the *set_min_delay* command.

observe_pin

Specifies the (internal) observe pin name of an LSI Logic scan macro cell (LSI CTV only). This attribute is used by the *write_test* command. Set with the *set_attribute* command.

pin_direction

Returns the direction of a pin. The value can be *in*, *out*, *inout*, or *unknown*. This attribute cannot be set by the user. You can also use the *direction* attribute to get the direction of the pin.

pin_properties

Lists valid EDIF property values to be attached to different versions of the output pin. The EDIF property values correspond to different output emitter-follower resistance values on the output pin. For information about how to use attribute, see the *Library Compiler Reference Manual*. Set with the *set_attribute* command.

set_pin

Specifies the (internal) set pin name of an LSI Logic scan macro cell (LSI CTV only). This attribute is used by the *write_test* command. Set with the *set_attribute* command.

signal_type

Used to indicate that a pin or port is of a special type, such as a *clocked_on_also* port in a master/slave clocking scheme, or a *test_scan_in* pin for scan-test circuitry. Set with the *set_signal_type* command.

static_probability

A floating point number that specifies the percentage of time that the signal is in the logic 1 state; this information is used by the *report_power* command. If this attribute is not set, the *report_power* command uses 0.5 (the default) as the value, indicating that the signal is in the logic 1 state half the time. Set with the *set_switching_activity* command.

test_assume

A string that represents a constant logic value to be assumed for specified pins throughout test design rule checking, test pattern generation, and fault simulation by the *check_test*, *create_test_patterns*, and *fault_simulate* commands. "1", "one", or "ONE" specifies a constant value of logic 1; "0", "zero", or "ZERO" specifies a constant value of logic 0. Use the *report_test -assertions* command for a report on objects that have the *test_assume* attribute set. Set with the *set_test_assume* command.

test_dont_fault

Specifies pins not faulted during test pattern generation. If no command options are specified, this attribute is set for both stuck-at-0 and stuck-at-1 faults. Set with the *set_test_dont_fault* command.

test_initial

A string that represents an initial logic value to be assumed for specified pins at the start of test design rule checking and fault simulation by the *check_test* and *fault_simulate* commands. "1", "one", or "ONE" specifies an initial value of logic 1; "0", "zero", or "ZERO" specifies an initial value of logic 0. Use the *report_test -assertions* command for a report on objects that have the *test_initial* attribute set. Set with the *set_test_initial* command.

test_isolate

Indicates that the specified sequential cells, pins, or ports are to be logically isolated and considered untestable during test design rule checking by the *check_test* command. When this attribute is set on a cell, it is also placed on all pins of that cell. Do not set this attribute on a hierarchical cell. Use the *report_test -assertions* command for a report on isolated objects. Set with the *set_test_isolate* command.

Note: Setting this attribute suppresses the warning messages associated with the isolated objects.

test_require

Specifies a constant, fixed logic value that a pin is required to have during scan test vector generation. The pin maintains the same value for each test vector generated. Use the *report_test -assertions* for a report on objects that have the *test_require* attribute set. Set with the *set_test_require* command.

test_routing_position

Specifies the preferred routing order of the scan-test signals of the identified cells. Set with the *set_test_routing_order* command.

switching_activity

A positive floating point number that specifies the switching activity, that is, the number of zero-to-one and one-to-zero transitions within a library time unit period. This information is used by the *report_power* command. If this attribute is not set, the *report_power* command uses the default value of 2*(static_probability)(1 - static_probability). The default value is scaled by any associated clock signal (if one is available). Set with the *set_switching_activity* command.

true_delay_case_analysis

Specifies a value to set all or part of an input vector for the *report_timing -true* and *report_timing -justify* commands. Allowed values are *0*, *1*, *r* (rise, X to 1), and *f* (fall, X to 0). Set with the *set_true_delay_case_analysis* command.

hold_uncertainty

Specifies a negative uncertainty from the edges of the ideal clock waveform. Affects all sequential cells in the transitive fanout of this pin. Set with the *set_clock_uncertainty -hold* command.

setup_uncertainty

Specifies a positive uncertainty from the edges of the ideal clock waveform. Affects all sequential cells in the transitive fanout of this pin. Set with the *set_clock_uncertainty -setup* command.

pg_pin_weight

A float attribute representing the PG pin weight. The PG pin weight should be between 0.1 and 25.0, and the fraction precision is 0.1. For example, 0.11 will be cut off as 0.1.

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",

Feedback

"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" } }
```

See Also

- get_attribute
- remove_attribute
- set_attribute
- attributes

---

placement_blockage_attributes

Contains attributes related to placement blockage.

Description

Contains attributes related to placement blockage.

You can use *get_attribute* to determine value of an attribute, and use *report_attribute* to get a report of all attributes on specified object. Specified with *list_attribute -class placement_blockage -application*, the definition of attributes can be listed.

Placement Blockage Attributes

Specifies the affects of a placement blockage, which is placement.

The data type of *affects* is string.

This attribute is read-only.

Specifies area of a placement blockage.

The data type of *area* is float.

Feedback

This attribute is read-only.

Specifies the bounding-box of a placement blockage. The *bbox* is represented by a *rectangle*.

The format of a *rectangle* specification is {{llx lly} {urx ury}}, which specifies the lower-left and upper-right corners of the rectangle.

The data type of *bbox* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies the percentage blockage for a partial blockage.

The data type of *blocked_percentage* is integer.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies Milkyway design ID in which a placement blockage object is located.

The data type of *cell_id* is integer.

This attribute is read-only.

Specifies layer name of a placement blockage.

The data type of *layer* is string.

Its valid values are:

- **`PlaceBlockage`**
- **`SoftPlaceBlk`**
- **`pinBlockage`**
- **`MacroBlockage`**
- **`PartialPlaceBlk`**

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies name of a placement blockage object.

The data type of *name* is string.

Feedback

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies object class name of a placement blockage, which is *placement_blockage*.

The data type of *object_class* is string.

This attribute is read-only.

Specifies object ID in Milkyway design file.

The data type of *object_id* is integer.

This attribute is read-only.

Specifies the layers for which routing to pins are blocked. This attribute only applies on pin blockage type.

The data type of *pin_blockage_layers* is string.

This attribute is read-only.

Specifies type of a placement blockage.

The data type of *type* is string.

Its valid values are:

- **`hard`**
- **`soft`**
- **`pin`**
- **`hard_macro`**
- **`partial`**

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

The attribute is true if the placement blockage has the property "is_reserved_placement_area".

The attribute is read-only.

no_register is a writable boolean attribute. If true, it specifies that register cells are prohibited within the blockage. All other cells are allowed.

Feedback

buffer_only is a writable boolean attribute. If true, it specifies that non-buffer cells are prohibited within the blockage. Buffers and inverters are allowed.

no_rp_group is a writable boolean attribute. If true, it specifies that rp groups are prohibited within the blockage. All other cells are allowed.

category_name is a writable string attribute. If set, it specifies the name of a boolean user-defined attribute. Use this attribute to specify which cells or references are prohibited within the blockage. See *create_placement_blockage* for details.

no_register, *buffer_only*, *no_rp_group* and *category_name* are mutually exclusive, i.e. they may not be used together on the same placement blockage. They may only be used on partial type blockages. See *create_placement_blockage* for more information.

buffer_only and *category_name* require the *redefined blockage behavior* feature, enabled as follows:

set placer_enable_redefined_blockage_behavior true

See Also

- create_placement_blockage
- get_attribute
- list_attributes
- report_attribute
- set_attribute
- placer_enable_redefined_blockage_behavior

---

plan_group_attributes

Contains attributes related to plan group.

Description

Contains attributes related to plan group.

You can use *get_attribute* to determine value of an attribute, and use *report_attribute* to get a report of all attributes on specified object. Specified with *list_attribute -class plan_group -application*, the definition of attributes can be listed.

Plan Group Attributes

Specifies the *height*:*width* ratio of a plan group.

The data type of *aspect_ratio* is double.

This attribute is read-only.

Specifies the bounding-box of a plan group. The *bbox* is represented by a *rectangle*.

The format of a *rectangle* specification is {{llx lly} {urx ury}}, which specifies the lower-left and upper-right corners of the rectangle.

The data type of *bbox* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies the lower-left corner of the bounding-box of a plan group.

The *bbox_ll* is represented by a *point*. The format of a *point* specification is {x y}.

You can get the *bbox_ll* of a plan group, by accessing the first element of its *bbox*.

The data type of *bbox_ll* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies x coordinate of the lower-left corner of the bounding-box of a plan group.

The data type of *bbox_llx* is double.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies y coordinate of the lower-left corner of the bounding-box of a plan group.

The data type of *bbox_lly* is double.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies the upper-right corner of the bounding-box of a plan group.

The *bbox_ur* is represented by a *point*. The format of a *point* specification is {x y}.

You can get the *bbox_ur* of a plan group, by accessing the second element of its *bbox*.

The data type of *bbox_ur* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies x coordinate of the upper-right corner of the bounding-box of a plan group.

The data type of *bbox_urx* is double.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies y coordinate of the upper-right corner of the bounding-box of a plan group.

The data type of *bbox_ury* is double.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies bottom-side width of interior paddings.

The data type of *bottom_padding* is double.

This attribute is read-only. You can use *create_fp_plan_group_padding* to set its value.

Specifies bottom-side width of exterior paddings.

The data type of *bottom_padding_external* is double.

This attribute is read-only. You can use *create_fp_plan_group_padding* to set its value.

Specifies bottom-side widths of signal shielding inside a plan group.

The data type of *bottom_shielding* is string.

This attribute is read-only. You can use *create_fp_block_shielding* to set its value.

Specifies bottom-side widths of signal shielding outside a plan group.

The data type of *bottom_shielding_external* is string.

This attribute is read-only. You can use *create_fp_block_shielding* to set its value.

Specifies color to draw a plan group and its associated instances.

The data type of *color* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies that a plan group is in a fixed location, and the shaping will ignore it.

The data type of *is_fixed* is boolean.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies that a plan group contains only the interface netlist.

The data type of *is_on_demand_netlist* is boolean.

This attribute is read-only.

Feedback

Specifies left-side width of interior paddings.

The data type of *left_padding* is double.

This attribute is read-only. You can use *create_fp_plan_group_padding* to set its value.

Specifies left-side width of exterior paddings.

The data type of *left_padding_external* is double.

This attribute is read-only. You can use *create_fp_plan_group_padding* to set its value.

Specifies left-side widths of signal shielding inside a plan group.

The data type of *left_shielding* is string.

This attribute is read-only. You can use *create_fp_block_shielding* to set its value.

Specifies left-side widths of signal shielding outside a plan group.

The data type of *left_shielding_external* is string.

This attribute is read-only. You can use *create_fp_block_shielding* to set its value.

Specifies name of the hierarchical cell associated with a plan group.

The data type of *logic_cell* is string.

This attribute is read-only.

Specifies name of a plan group object.

The data type of *name* is string.

This attribute is read-only.

Specifies number of hard macro cells inside a plan group.

The data type of *number_of_hard_macro* is integer.

This attribute is read-only.

Specifies number of pins on the hierarchical cell associated with a plan group.

The data type of *number_of_pin* is integer.

This attribute is read-only.

Specifies number of standard cells inside a plan group.

The data type of *number_of_standard_cell* is integer.

This attribute is read-only.

Specifies number of standard cells inside an ODL plan group.

Feedback

The data type of *number_of_odl_standard_cell* is integer.

This attribute is read-only.

Specifies object class name of a plan group, which is *plan_group*.

The data type of *object_class* is string.

This attribute is read-only.

Specifies point list of a plan group's boundary.

The data type of *points* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies right-side width of interior paddings.

The data type of *right_padding* is double.

This attribute is read-only. You can use *create_fp_plan_group_padding* to set its value.

Specifies right-side width of exterior paddings.

The data type of *right_padding_external* is double.

This attribute is read-only. You can use *create_fp_plan_group_padding* to set its value.

Specifies right-side widths of signal shielding inside a plan group.

The data type of *right_shielding* is string.

This attribute is read-only. You can use *create_fp_block_shielding* to set its value.

Specifies right-side widths of signal shielding outside a plan group.

The data type of *right_shielding_external* is string.

This attribute is read-only. You can use *create_fp_block_shielding* to set its value.

Specified target utilization set on a plan group.

The data type of *target_utilization* is double.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies top-side width of interior paddings.

The data type of *top_padding* is double.

This attribute is read-only. You can use *create_fp_plan_group_padding* to set its value.

Specifies top-side width of exterior paddings.

The data type of *top_padding_external* is double.

This attribute is read-only. You can use *create_fp_plan_group_padding* to set its value.

Specifies top-side widths of signal shielding inside a plan group.

The data type of *top_shielding* is string.

This attribute is read-only. You can use *create_fp_block_shielding* to set its value.

Specifies top-side widths of signal shielding outside a plan group.

The data type of *top_shielding_external* is string.

This attribute is read-only. You can use *create_fp_block_shielding* to set its value.

Specifies the ratio of total area size of associated instances to the area of a plan group.

The data type of *utilization* is double.

This attribute is read-only.

Specifies the ratio of total area size of associated instances to the area of an ODL plan group.

The data type of *odl_utilization* is float.

This attribute is read-only.

Specifies that a plan group is multiply instantiated module (MIM).

The data type of *is_mim* is Boolean.

This attribute is read-only.

Specifies that a plan group is a master instance among a set of multiply instantiated modules (MIMs).

The data type of *is_mim_master_instance* is Boolean.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies the name of the master for a multiply instantiated module (MIM) plan group.

The data type of *mim_master_name* is string.

This attribute is read-only.

Specifies the orientation of a MIM plan group.

The allowable orientations are N, S, FN, FS.

The data type of *mim_orientation* is string.

Feedback

This attribute is read-only.

See Also

- get_attribute
- list_attributes
- report_attribute
- set_attribute

---

port_attributes

Contains attributes that can be placed on a port.

Description

Contains attributes that can be placed on a port.

There are a number of commands used to set attributes. Most attributes, however, can be set with the *set_attribute* command. If the attribute definition specifies a *set* command, use it to set the attribute. Otherwise, use *set_attribute*. If an attribute is "read-only," the user cannot set it.

To determine the value of an attribute, use the *get_attribute* command. To remove attributes, use the *remove_attribute* command.

For a more detailed explanation of an attribute, refer to the manual pages of the appropriate *set* command. For information on all attributes, refer to the *attributes* manual page.

Port Attributes

bus_name

If the port is bussed, returns the full hierarchical name of the port bus. This attribute is *read only* and cannot be set by the user.

connection_class

A string that specifies the connection class label to be attached to a port or to a list of ports. *compile*, *insert_pads*, and *insert_dft* will connect only those loads and drivers that have the same connection class label. The labels must match those in the library of components for the design, and must be separated by a space. The labels *universal* and *default* are reserve.d *universal* indicates that the port can connect with any other load or driver. *default* is assigned to any ports that do not have a connection class already assigned. Set with *set_connection_class*.

dont_touch_network

When a design is optimized, *compile* assigns *dont_touch* attributes to all cells and nets in the transitive fanout of *dont_touch_network* clock objects. The *dont_touch* assignment stops at the boundary of storage elements. An element is recognized as storage only if it has setup or hold constraints. Set with *set_dont_touch_network*.

driven_by_logic_one

Specifies that input ports are driven by logic one. *compile* uses this information to create smaller designs. After optimization, a port connected to logic one usually does not drive anything inside the optimized design. Set with *set_logic_one*.

driven_by_logic_zero

Specifies that input ports are driven by logic zero. *compile* uses this information to create smaller designs. After optimization, a port connected to logic zero usually does not drive anything inside the optimized design. Set with *set_logic_zero*.

driving_cell_dont_scale

When *true*, the transition time on the port using the driving cell is not scaled. Otherwise the transition time will be scaled by operating condition factors. Set with *set_driving_cell*.

driving_cell_fall

A string that names a library cell from which to copy fall drive capability to be used in fall transition calculation for the port. Set with *set_driving_cell*.

driving_cell_from_pin_fall

A string that names the driving_cell_fall input pin to be used to find timing arc fall drive capability. Set with *set_driving_cell*.

driving_cell_from_pin_rise

A string that names the driving_cell_rise input pin to be used to find timing arc rise drive capability. Set with *set_driving_cell*.

driving_cell_library_fall

A string that names the library in which to find the *driving_cell_fall*. Set with *set_driving_cell*.

driving_cell_library_rise

A string that names the library in which to find the *driving_cell_rise*. Set with *set_driving_cell*.

driving_cell_multiply_by

A floating point value by which to multiply the transition time of the port marked with this attribute. Set with *set_driving_cell*.

driving_cell_pin_fall

A string that names the driving_cell_fall output pin to be used to find timing arc fall drive capability. Set with *set_driving_cell*.

driving_cell_pin_rise

A string that names the driving_cell_rise output pin to be used to find timing arc rise drive capability. Set with *set_driving_cell*.

driving_cell_rise

A string that names a library cell from which to copy rise drive capability to be used in rise transition calculation for the port. Set with *set_driving_cell*.

fall_drive

Specifies the drive value of high to low transition on input or inout ports. Set with *set_drive*.

fanout_load

Specifies the fanout load on output ports. Set with *set_fanout_load*.

is_bussed

Returns true if the port is part of a bussed port. This attribute is *read only* and cannot be set by the user.

load

Specifies the load value on ports. The total load on a net is the sum of all the loads on pins, ports, and wires associated with that net. Set with *set_load*.

max_capacitance

A floating point number that sets the maximum capacitance value for input, output, or bidirectional ports; or for designs. The units must be consistent with those of the technology library used during optimization. Set with *set_max_capacitance*.

max_fall_delay

A floating point value that specifies the maximum falling delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_max_delay*.

max_fanout

Specifies the maximum fanout load for the net connected to this port. *compile* ensures that the fanout load on this net is less than the specified value. Set with *set_max_fanout*.

max_rise_delay

A floating point value that specifies the maximum rising delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_max_delay*.

max_time_borrow

A floating point number that establishes an upper limit for time borrowing; that is, it prevents the use of the entire pulse width for level-sensitive latches. Units are those used in the technology library. Set with *set_max_time_borrow*.

max_transition

Specifies the maximum transition time for the net connected to this port. *compile* ensures that the transition time on this net is less than the specified value. Set with *set_max_transition*.

min_capacitance

A floating point number that sets the minimum capacitance value for input or bidirectional ports. The units must be consistent with those of the technology library used during optimization. Set with *set_min_capacitance*.

min_fall_delay

A floating point value that specifies the minimum falling delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_min_delay*.

min_rise_delay

A floating point value that specifies the minimum rising delay on ports, clocks, pins, cells, or on paths between such objects. Set with *set_min_delay*.

model_drive

A non-negative floating point number that specifies the estimated drive value on ports in terms of standard drives of the current technology library. Set with *set_model_drive*.

model_load

A non-negative floating point number that specifies the estimated load value on ports in terms of standard loads of the current technology library. Set with *set_model_load*.

op_used_in_normal_op

Specifies that a scan-out port is also used in normal operation (system mode). This attribute is used by the *insert_dft* command.

output_not_used

Determines that an output port is unconnected. Used by *compile* to create smaller designs because the logic that drives an unconnected output port might not need to be maintained. After a design with an unconnected output port is compiled, the port is usually not driven by anything inside the design. Set with *set_unconnected*.

pad_location

A string value that specifies the Xilinx pad location (pin number) to be assigned to a port. Setting a *pad_location* attribute on a port causes the Synopsys XNF writer to indicate in the XNF netlist that this port has the pad location given by the value of the *pad_location* attribute. No checks are performed to verify that the specified location is valid; for valid pad locations, refer to the Xilinx *XC4000 Databook*. Set with *set_attribute*.

port_direction

Returns the direction of a port. The value can be *in*, *out*, *inout*, or *unknown*. This attribute cannot be set by the user. You can also use the *direction* attribute to get the direction of the port.

rise_drive

Specifies the drive value of low to high transition on input or inout ports. Set with *set_drive*.

signal_index

Used to enumerate different ports with the same signal type (for example, scan-in ports for a design with multiple scan chains). Set with *set_signal_type*.

signal_type

Used to indicate that a port is of a "special" type, such as a "clocked_on_also" port in a master/slave clocking scheme, or a "test_scan_in" pin for scan-test circuitry. Set with *set_signal_type*.

static_probability

A floating point number that specifies the percentage of time that the signal is in the logic 1 state. This information is used by *report_power*. If this attribute is not set, *report_power* will use the default value of 0.5, indicating that the signal is in the logic 1 state half the time. Set with *set_switching_activity*.

test_dont_fault

If set, ports are not faulted during test pattern generation. If no command options are specified, this attribute is set for both "stuck-at-0" and "stuck-at-1" faults. Set with *set_test_dont_fault*.

test_hold

Specifies a fixed, constant logic value at a port during test generation. Set with *set_test_hold*.

test_isolate

Indicates that the specified sequential cells, pins, or ports are to be logically isolated and considered untestable during test design rule checking by *check_test*. When this attribute is set on a cell, it is also placed on all pins of that cell. Do not set this attribute on a hierarchical cell. Use *report_test -assertions* for a report on isolated objects. Set with *set_test_isolate*.

Note: Setting this attribute suppresses the warning messages associated with the isolated objects.

toggle_rate

A positive floating point number that specifies the toggle rate; that is, the number of zero-to-one and one-to-zero transitions within a library time unit period. This information is used by *report_power*. If this attribute is not set, *report_power* will use the default value of 2*(static_probability)(1 - static_probability). The default will be scaled by any associated clock signal (if one is available). Set with *set_switching_activity*.

true_delay_case_analysis

Specifies a value to set all or part of an input vector for *report_timing -true* and *report_timing -justify*. Allowed values are *0*, *1*, *r* (rise, X to 1), and *f* (fall, X to 0). Set with *set_true_delay_case_analysis*.

is_diode

Tells if the port is a diode port.

is_bus

Tells if the port is a bus port.

This attribute is read-only and cannot be modified.

hold_uncertainty

Specifies a negative uncertainty from the edges of the ideal clock waveform. Affects all sequential cells in the transitive fanout of this port. Set with *set_clock_uncertainty -hold*.

Feedback

setup_uncertainty

Specifies a positive uncertainty from the edges of the ideal clock waveform. Affects all sequential cells in the transitive fanout of this port. Set with *set_clock_uncertainty -setup*.

pg_pin_weight

A float attribute representing the PG pin weight. It should be between 0.1 and 25.0. And the fraction precision is 0.1, e.g. 0.11 will be cut off as 0.1.

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".

See Also

- get_attribute
- remove_attribute
- set_attribute
- attributes

---

power_attributes

Lists the predefined Power Compiler attributes.

Description

Attributes are properties assigned to objects such as nets, cells, pins and designs, and describe design features to be considered during optimization. A subset of attributes are specific to Power Compiler; they are only available when a Power Compiler license exists.

The Power Compiler attributes are "read-only": they cannot be set by the user. To determine the value of an attribute, use the *get_attribute* command. For information on all attributes, refer to the *attributes* manual page.

The Power Compiler attributes are grouped into the following categories:

```
- cell
- pin
- design
```

Definitions for these attributes are provided in the subsections that follow.

Cell Attributes

is_clock_gate

true if the cell is a clock gate. If the clock gating logic is encapsulated in a hierarchical clock gate wrapper, this attribute will only return *true* when applied to the hierarchical instance.

is_icg

true if the cell is an integrated clock gate (ICG). This attribute can only return *true* when applied to leaf cells. If the ICG cell is encapsulated in a hierarchical clock gate wrapper, this attribute will return *false* when applied to the hierarchical instance.

is_gicg

true if the cell is a generic integrated clock gate (GICG). Since GICGs cannot be encapsulated in a hierarchical clock gate wrapper, this attribute can only return *true* when applied to leaf cells.

is_latch_based_clock_gate

true if the cell is a latch-based clock gate.

is_latch_free_clock_gate

true if the cell is a latch-free clock gate.

is_positive_edge_clock_gate

true if the cell is a positive edge clock gate.

is_negative_edge_clock_gate

true if the cell is a negative edge clock gate.

clock_gate_has_precontrol

true if the cell is a clock gate with (pre-latch) control point.

clock_gate_has_postcontrol

true if the cell is a clock gate with (post-latch) control point.

clock_gate_has_observation

true if the cell is a clock gate with observation point.

is_clock_gated

true if the cell is a clock gated register or clock gate.

clock_gating_depth

The number of clock gates on the clock path to this cell; -1 if the cell is not a clock gate or register.

clock_gate_level

The number of clock gates on the longest clock branch in the fanout of this cell; -1 if not a clock gate.

clock_gate_fanout

The number of registers and clock gates in the direct fanout of the clock gate; -1 if not a clock gate.

clock_gate_register_fanout

The number of registers in the direct fanout of the clock gate; -1 if not a clock gate.

clock_gate_multi_stage_fanout

The number of clock gates in the direct fanout of the clock gate; -1 if not a clock gate.

clock_gate_transitive_register_fanout

The number of register in the transitive fanout of the clock gate; -1 if not a clock gate.

clock_gate_module_fanout

The number of modules in the local fanout of the clock gate; -1 if not a clock gate.

is_operand_isolator

true if the cell is an operand isolation cell.

is_isolated_operator

true if the cell is an operator that was isolated with operand isolation.

operand_isolation_style

Stores the operand isolation style of the isolation cell or isolated operator.

Pin Attributes

is_clock_gate_enable_pin

true if the pin is a clock gate enable input.

Feedback

is_clock_gate_clock_pin

true if the pin is a clock gate clock input.

is_clock_gate_output_pin

true if the pin is a gated-clock output of a clock gate.

is_clock_gate_test_pin

true if the pin is a clock gate scan-enable or test-mode input.

is_clock_gate_observation_pin

true if the pin is a clock gate observation point.

is_operand_isolation_control_pin

true if the pin is the control pin of an operand isolation cell.

is_operand_isolation_data_pin

true if the pin is the data input of an operand isolation cell.

is_operand_isolation_output_pin

true if the pin is the data output of an operand isolation cell.

Design Attributes

is_clock_gating_design

true if the design is a clock gating design.

is_clock_gating_observability_design

true if the design is a clock gating observability design.

See Also

- get_attribute
- attributes

---

power_domain_attributes

Describes the attributes related to power domains.

Description

Describes the attributes related to power domains. These attributes are defined only in UPF mode.

You can use the *get_attribute* command to determine value of an attribute, and use the *report_attribute* command to get a report of all attributes on a specified power domain. To see all power domain attributes, use the *list_attribute -class power_domain -application* command.

Power Domain Attributes

cell_id

Specifies Milkyway design ID in which a power domain object is located.

This attribute is read-only.

name

Specifies name of a power domain object.

This attribute is read-only.

full_name

Specifies full name of a power domain object.

This attribute is read-only.

object_class

Specifies object class name of a power domain object, which is *power_domain*.

This attribute is read-only.

object_id

Specifies object ID in Milkyway design file.

This attribute is read-only.

within_block_abstraction

Specifies whether the power domain is part of the block abstraction.

The data type of *within_block_abstraction* is Boolean.

This attribute is read-only and cannot be modified.

within_ilm

Specifies whether the power domain is part of an ILM.

The data type of *within_ilm* is Boolean.

This attribute is read-only and cannot be modified.

Feedback

See Also

- get_attribute
- list_attributes
- report_attribute

---

power_switch_attributes

Describes the attributes related to power switches.

Description

This man page described the attributes related to power switches. These attributes are defined only in UPF mode.

You can use the *get_attribute* command to determine value of an attribute, and use the *report_attribute* command to get a report \ of all the attributes on a specified power switch. To see all the power-switch attributes, use the *list_attribute -class power_switch -application* command.

Power Switch Attributes

cell_id

Specifies Milkyway design ID in which a power switch object is located.

This attribute is read-only.

name

Specifies name of a power switch object.

This attribute is read-only.

full_name

Specifies full name of a power switch object.

This attribute is read-only.

object_class

Specifies object class name of a power switch object, which is *power_switch*.

This attribute is read-only.

object_id

Specifies object ID in Milkyway design file.

This attribute is read-only.

Feedback

within_block_abstraction

Specifies whether the power switch is part of the block abstraction.

The data type of *wthin_block_abstraction* is boolean.

This attribute is read-only and cannot be modified.

within_ilm

Specifies whether the power switch is part of an ILM.

The data type of *within_ilm* is boolean.

This attribute is read-only and cannot be modified.

See Also

- get_attribute
- list_attributes
- report_attribute

---

r

---

read_only_attributes

Contains informational attributes, which the user cannot set.

Description

Contains informational attributes. A "read-only" attribute cannot be set by the user.

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

For information on all attributes, refer to the *attributes* manual page.

Read-only Attributes

design_type

Indicates the current state of the design and has the value *fsm* (finite state machine), *pla* (programmable logic array), *equation* (Boolean logic), or *netlist* (gates). This attribute cannot be set by the user.

is_black_box

true if the reference is not yet linked to a design. This attribute cannot be set by the user.

is_combinational

true if all cells of a design and all designs in its hierarchy are combinational. A cell is combinational if it is non-sequential or non-tristate and all of its outputs compute a combinational logic function. The *report_lib* command will report such a cell as not a black-box. This attribute cannot be set by the user.

is_dw_subblock

true if the object (a cell, a reference, or a design) is a DW subblock that was automatically elaborated. This attribute cannot be set by the user.

Note: DW subblocks that are manually elaborated will not have this attribute.

is_hierarchical

true if the design contains leaf cells or other levels of hierarchy. This attribute is read-only and cannot be set by the user.

is_mapped

true if all the non-hierarchical cells of a design are mapped to cells in a technology library. This attribute cannot be set by the user.

is_sequential

true if any cells of a design or designs in its hierarchy are sequential. A cell is sequential if it is not combinational. This attribute cannot be set by the user.

is_test_circuitry

Set by *insert_dft* on the scan cells and nets added to a design during the addition of test circuitry. This attribute cannot be set by the user.

is_synlib_module

true if the object (a cell, a reference, or a design) refers to an unmapped module reference or if the object is (or refers to) a design that was automatically elaborated from a synlib module or a synlib operator. This attribute cannot be set by the user.

Note: synlib modules that are manually elaborated will not have this attribute.

is_synlib_operator

true if the object (a cell or a reference) is a synthetic library operator reference. This attribute cannot be set by the user.

is_unmapped

true if any of the cells are not linked to a design or mapped to a technology library. This attribute cannot be set by the user.

Feedback

pin_direction

Direction of a pin. Value can be *in*, *out*, *inout*, or *unknown*. This attribute cannot be set by the user.

port_direction

Direction of a port. Value can be *in*, *out*, *inout*, or *unknown*. This attribute cannot be set by the user.

ref_name

The reference name of a cell. This attribute cannot be set by the user.

hdl_hier

The original (RTL) hierarchy of a cell or a pin's parent cell. Returns "-" if there is no RTL information (e.g. if the cell is synthethized and has no RTL origin). This attribute cannot be set by the user.

See Also

- get_attribute
- attributes

---

reference_attributes

Contains attributes that can be placed on a reference.

Description

Contains attributes that can be placed on a reference.

Several commands exist that can be used to set attributes; however, most attributes can be set by using the *set_attribute* command. If the attribute definition specifies a *set* command, use it to set the attribute; otherwise, use *set_attribute*. If an attribute is read-only, the you cannot set it.

To determine the value of an attribute, use the *get_attribute* command. To remove attributes, use the *remove_attribute* command.

For a more detailed explanation of an attribute, see the manual page of the appropriate *set* command. For information on all attributes, refer to the *attributes* manual page.

Reference Attributes

dont_touch

Specifies that designs linked to a reference with this attribute are excluded from optimization. Valid values are true (the default) or false. Designs linked

to a reference by using the *dont_touch* attribute set to true are not modified or replaced during compile. Set this by using the *set_dont_touch* attribute.

is_black_box

This is set to true if the reference is not yet linked to a design. This attribute is read-only and you cannot set it.

is_combinational

This is set to true if all the cells of the referenced design are combinational. A cell is combinational if it is nonsequential or non-tristate and all of its outputs compute a combinational logic function. The *report_lib* command reports such a cell as not a black-box. This attribute is read-only and you cannot set it.

is_dw_subblock

This is set to true if the object (a cell, a reference, or a design) is a DesignWare subblock that was automatically elaborated. This attribute is read-only and you cannot set it.

Note: DesignWare subblocks that are manually elaborated do not have this attribute.

is_hierarchical

This is set to true if the design contains leaf cells or other levels of hierarchy. This attribute is read-only and you cannot set it.

is_mapped

This is set to true if the reference is linked to a design, and all the non-hierarchical cells of the referenced design are mapped to cells in a technology library. This attribute is read-only and you cannot set it.

is_sequential

This is set to true if all the cells of the referenced design are sequential. A cell is sequential if it is not combinational (if any of its outputs depend on previous inputs). This attribute is read-only and you cannot set it.

is_synlib_module

This is set to true if the object (a cell, a reference, or a design) refers to an unmapped module reference, or the object is (or refers to) a design that was automatically elaborated from a synlib module or a synlib operator. This attribute is read-only and you cannot set it.

Note: synlib modules that are manually elaborated do not have this attribute.

Feedback

is_synlib_operator

This is set to true if the object (a cell or a reference) is a synthetic library operator reference. This attribute is read-only and you cannot set it.

is_unmapped

This is set to true if any of the non-hierarchical cells of the referenced design are not mapped to cells in a technology library, or the reference is not yet linked to a design. This attribute is read-only and you cannot set it.

scan

When *true*, specifies that cells of the referenced design are always replaced by equivalent scan cells during insert_dft. When false, cells are not replaced. Set by using the *set_scan_replacement*.

scan_chain

Includes the specified cells of the referenced design in the scan-chain whose index is the value of this attribute.

ungroup

Specifies that all designs linked to a reference with this attribute are ungrouped (levels of hierarchy represented by these design cells are removed) during compile. Set by using the *set_ungroup* command.

See Also

- get_attribute
- insert_dft
- remove_attribute
- set_attribute
- set_scan_replacement
- attributes

---

route_guide_attributes

Contains attributes related to route guide.

Description

Contains attributes related to route guide.

You can use *get_attribute* to determine value of an attribute, and use *report_attribute* to get a report of all attributes on specified object. Specified with *list_attribute -class route_guide -application*, the definition of attributes can be listed.

Route Guide Attributes

Specifies the affects of a route guide, which is route.

The data type of *affects* is string.

This attribute is read-only.

Specifies area of a route guide.

The data type of *area* is float.

This attribute is read-only.

Specifies the bounding-box of a route guide. The *bbox* is represented by a *rectangle*.

The format of a *rectangle* specification is {{llx lly} {urx ury}}, which specifies the lower-left and upper-right corners of the rectangle.

The data type of *bbox* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies Milkyway design ID in which a route guide object is located.

The data type of *cell_id* is integer.

This attribute is read-only.

Specifies the layers on which router will follow design boundary blockage rules.

The data type of *design_boundary_blockage* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies the coloring information of the route_guide. This information is needed in the double patterning flow.

The data type of *double_pattern_mask_constraint* is string.

The valid values can be: any_mask, mask1_soft, mask1_hard, mask2_soft, mask2_hard and same_mask.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies the horizontal track utilization for the route guide.

The data type of *horizontal_track_utilization* is integer.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies layer name on which a route guide is.

The data type of *layer* is string.

This attribute is read-only.

Specifies layer number on which a route guide is.

The data type of *layer_number* is integer.

This attribute is read-only.

Specifies the coloring information of the route_guide. This information is needed in the multiple patterning flow.

The data type of *multiple_pattern_mask_constraint* is string.

The valid values can be: any_mask, mask1_soft, mask1_hard, mask2_soft, mask2_hard, mask3_soft, mask3_hard and same_mask.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies name of a route guide object.

The data type of *name* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies the layers that cannot contain preroutes.

The data type of *no_preroute_layers* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies the layers that cannot contain signals.

The data type of *no_signal_layers* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies object class name of a route guide, which is *route_guide*.

The data type of *object_class* is string.

This attribute is read-only.

Specifies object ID in Milkyway design file.

The data type of *object_id* is integer.

This attribute is read-only.

Specifies geometry type of a route guide, which can be RECTANGLE or POLYGON.

The data type of *object_type* is string.

This attribute is read-only.

Specifies point list of a route guide's boundary.

The data type of *points* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies the layers that cannot make nonPreferredDirection wires.

The data type of *preferred_direction_only_layers* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies whether this route guide should be repaired as a single sbox when there is a difficult violation on a prerouted wire or inside a large macro.

The data type of *repair_as_single_sbox* is boolean.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Route guides with the same route guide group id belong to the same route guide group.

Specifies whether to switch the preferred direction for the route guide.

The data type of *switch_preferred_direction* is boolean.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies the layers on which single layer routing should be encouraged.

The data type of *switch_preferred_direction_layers* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Feedback

Specifies the layers on which horizontal_track_utilization or vertical_track_utilization had been set.

The data type of *track_utilization_layers* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies the affects of a route guide, which is route.

The data type of *type* is string.

This attribute is read-only.

Specifies the vertical track utilization for the route guide.

The data type of *vertical_track_utilization* is integer.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies whether zero minimum spacing is allowed for the route guide.

The data type of *zero_min_spacing* is boolean.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

See Also

- get_attribute
- list_attributes
- report_attribute
- set_attribute

---

routing_corridor_attributes

Contains attributes related to routing corridor.

Description

Contains attributes related to routing corridor.

You can use *get_attribute* to determine the value of an attribute, and use *report_attribute* to get a report of all attributes on the specified object. Specified with *list_attribute -class routing_corridor -application*, the definition of attributes can be listed.

Feedback

Routing Corridor Attributes

Specifies name of a routing corridor object.

The data type of *name* is string.

This attribute is read-only.

Specifies the nets associated with the routing corridor.

The data type of *nets* is collection.

This attribute is read-only.

Specifies object class name of a routing corridor, which is *routing_corridor*.

The data type of *object_class* is string.

This attribute is read-only.

See Also

- get_attribute
- list_attributes
- report_attribute
- set_attribute

---

routing_corridor_shape_attributes

Attributes related to routing corridor shapes.

Description

This man page describes the attributes related to routing corridor shapes.

To list the definitions of the routing corridor shape attributes, use the *list_attributes -application -class routing_corridor_shape* command.

To determine the value of an attribute, use the *get_attribute* command. To get a report of all attributes on a specified object, use the *report_attribute* command.

Routing Corridor Shape Attributes

Specifies the bounding-box of a routing corridor shape, which is represented by a rectangle.

The format of a rectangle specification is {{llx lly} {urx ury}}, which specifies the lower-left and upper-right corners of the rectangle.

Feedback

The data type of *bbox* is string.

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

Specifies layer name for the maximum layer constraint for this shape.

The data type of *max_layer* is string.

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

Specifies layer name for the minimum layer constraint for this shape.

The data type of *min_layer* is string.

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

Specifies the name of a routing corridor shape.

The data type of *name* is string.

This attribute is read-only.

Specifies object class name of a routing corridor shape, which is *routing_corridor_shape*.

The data type of *object_class* is string.

This attribute is read-only.

See Also

- get_attribute
- list_attributes
- report_attribute
- set_attribute

---

rp_group_attributes

Contains attributes that can be placed on a relative placement group.

Description

Contains attributes that can be placed on a relative placement group.

To set an attribute, use the command identified in the individual description of that attribute. If an attribute is read-only, you cannot set it.

To determine the value of an attribute, use the *get_attribute* command. To remove attributes, use the *remove_attribute* command.

For a more detailed explanation of an attribute, see the man page of the appropriate *set* command. For information about all attributes, see the *attributes* man page.

RP Group Attributes

alignment

Specifies the default alignment method to use when placing leaf cells and relative placement groups in the specified relative placement groups. If you do not specify this option, the tool uses bottom-left alignment.

You can set this attribute by using the *set_rp_group_options* command.

allow_non_rp_cells

Specifies that the hard keepout in the relative placement group should be overlapped with tap cells if needed. By default, the value is *false*.

You can set this attribute by using the *set_rp_group_options* command.

anchor_corner

Specifies the corner for the anchor point that is set by using the *-x_offset* and *-y_offset* options.

Valid values are *bottom-left* (the default), *bottom-right*, *top-left*, *top-right*, and *rp-location*.

If you specify *bottom-left*, the anchor point corner is the lower-left corner of the relative placement group.

If you specify *bottom-right*, the anchor point corner is the lower-right corner of the relative placement group.

If you specify *top-left*, the anchor point corner is the upper-left corner of the relative placement group.

If you specify *top-right*, the anchor point corner is the upper-right corner of the relative placement group.

If you specify *rp-location*, the anchor point corner is the starting location of the object at the row and column specified by the *-anchor_row* and *-anchor_column options.*

This option applies only to top-level relative placement groups and is ignored for hierarchical relative placement groups. When you specify *-anchor_corner bottom-right*, the relative placement group is anchored at the bottom-right corner

at the specified x- and y-coordinates during legalization. When many blockages are present, a slight deviation from the anchor point might occur.

You can set this attribute by using the *set_rp_group_options* command.

cell_orient_opt

A Boolean value that specifies if cell orientation optimization is done for the cells of the relative placement group for optimizing wire-length. This is a read-only attribute and cannot be modified by the user.

columns

An integer value that specifies the number of columns of the specified relative placement group.

This is a read-only attribute and cannot be modified by the user.

compress

A Boolean variable that specifies if compression is set in the horizontal direction to a relative placement group during placement. Setting this option places each row of a relative placement group without any gaps between leaf cells, lower-level hierarchical relative placement groups, or keepouts. Column alignment is not maintained when you use the *-compress* option.

You can set this attribute by using the *set_rp_group_options* command.

is_top

A Boolean value that specifies whether a relative placement group is the top.

This is a read-only attribute and cannot be modified by the user.

cts_option

Specifies how to treat the cells in the relative placement group during clock tree synthesis and clock tree optimization.

Valid values are *fixed_placement* (the default) and *size_only*.

If you specify *fixed_placement*, the cells in the relative placement group are treated as fixed during the *compile_clock_tree*, *optimize_clock_tree*, and *clock_opt* commands and cannot be sized or moved.

If you specify *size_only*, the cells in the relative placement group can only be sized. This option is applicable to the *compile_clock_tree*, *optimize_clock_tree*, and *clock_opt* commands.

You can set this attribute by using the *set_rp_group_options* command.

group_orient

A string that specifies the user-specified orientation that is set on the relative placement group.

Valid values are *default*, *N* (north), *S* (south), *FN* (flip-north), and *FS* (flip-south). The default is *N*.

You can set this attribute by using the *set_rp_group_options* command.

ignore

A Boolean value that specifies whether a relative placement group is ignored.

You can set this attribute by using the *set_rp_group_options* command.

move_effort

A string that specifies the move effort of relative placement group.

Valid values are *high*, *medium* (the default), and *low*.

You can set this attribute by using the *set_rp_group_options* command.

name

A string that specifies the full name of a relative placement group.

This is a read-only attribute and cannot be modified by the user.

pin_align_name

A string the specifies the name of pin on which the relative placement cells will be aligned.

You can set this attribute by using the *set_rp_group_options* command.

placement_type

A string that specifies the type of placement of a relative placement group.

You can set this attribute by using the *set_rp_group_options* command.

psynopt_option

Specifies the behavior of the relative placement cells of the specified relative placement group during the *psynopt* command.

You can set this attribute by using the *set_rp_group_options* command.

route_opt_option

Specifies the behavior of the relative placement cells of the specified relative placement group during the *route_opt* command.

You can set this attribute by using the *set_rp_group_options* command.

rows

An integer value that specifies the number of rows of a relative placement group.

This is a read-only attribute and cannot be modified by the user.

rp_height

A string that specifies the height of a relative placement group. It contains an estimated height before placement and the actual height if the relative placement group has gone through any kind of placement.

This is a read-only attribute and cannot be modified by the user.

rp_width

A string that specifies the width of a relative placement group. It contains an estimated width before placement and the actual width if the relative placement group has gone through any kind of placement.

This is a read-only attribute and cannot be modified by the user.

placed_orient

A string that specifies the placed orientation of a relative placement group.

Valid values are *N*, *S*, *FN*, and *FS*. If the relative placement group is not placed, it reports that the relative placement group is not placed.

This is a read-only attribute and cannot be modified by the user.

is_placed

A Boolean value that specifies whether a relative placement group is placed.

If the relative placement group is placed without any critical failures, it is set to *true*; otherwise, it is set to *false*.

This is a read-only attribute and cannot be modified by the user.

utilization

A floating point nonzero, positive value that specifies the area utilization of a relative placement group.

The default 1. The maximum value is 1.

You can set this attribute by using the *set_rp_group_options* command.

Feedback

x_offset

A floating point value that specifies the x-coordinate of a relative placement group's anchor point.

You can set this attribute by using the *set_rp_group_options* command.

y_offset

A floating point value that specifies the y-coordinate of a relative placement group's anchor point.

You can set this attribute by using the *set_rp_group_options* command.

anchor_row

An integer value that specifies the row of the object for which the rp-location anchor corner is specified.

You can set this attribute by using the *set_rp_group_options* command.

anchor_column

An integer value that specifies the column of the object for which the rp-location anchor corner is specified.

You can set this attribute by using the *set_rp_group_options* command.

S

shape_attributes

Attributes related to shapes.

Description

This man page describes attributes related to shapes.

To list the definitions of the shape attributes, use the *list_attribute -application -class shape* command.

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

To report all attributes on a specified object, use the *report_attribute* command.

Shape Attributes

Specifies the bounding-box of a shape.

The *bbox* attribute is represented by a rectangle. The format of a rectangle specification is {{llx lly} {urx ury}}, which specifies the lower-left and upper-right corners of the rectangle.

Feedback

The data type of *bbox* is string.

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

Specifies the lower-left corner of the bounding-box of a shape.

The *bbox_ll* attribute is represented by a point. The format of a point specification is {x y}.

You can get the *bbox_ll* attribute of a shape by accessing the first element of its *bbox* attribute.

The data type of *bbox_ll* is string.

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

Specifies the x-coordinate of the lower-left corner of the bounding-box of a shape.

The data type of *bbox_llx* is integer.

This attribute is read-only.

Specifies the y-coordinate of the lower-left corner of the bounding-box of a shape.

The data type of *bbox_lly* is integer.

This attribute is read-only.

Specifies the upper-right corner of the bounding-box of a shape.

The box_ur attribute is represented by a point. The format of a point specification is {x y}.

You can get the *bbox_ur* attribute of a shape by accessing the second element of its *bbox* attribute.

The data type of *bbox_ur* is string.

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

Specifies the x-coordinate of the upper-right corner of the bounding-box of a shape.

The data type of *bbox_urx* is integer.

This attribute is read-only.

Specifies the y-coordinate of the upper-right corner of the bounding-box of a shape.

The data type of *bbox_ury* is integer.

This attribute is read-only.

Specifies the Milkyway design ID in which a shape object is located.

Feedback

The data type of *cell_id* is integer.

This attribute is read-only.

Specifies the GDSII datatype number of a shape object.

The data type of *datatype_number* is integer.

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

Specifies the coloring information for a shape. This information is needed in the double-patterning flow.

The data type of *double_pattern_mask_constraint* is string.

Its valid values are

- *any_mask*
- *mask1_soft*
- *mask1_hard*
- *mask2_soft*
- *mask2_hard*
- *same_mask*

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

Specifies the alignment type for the end of a wire or path shape.

The data type of *endcap* is string.

Its valid values are

- *square_ends*
- *round_ends*
- *square_ends_by_half_width*
- *octagon_ends_by_half_width*

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

Specifies the via ladder is true or not.

The data type of *is_via_ladder* is boolean.

Feedback

This attribute is writable.

Specifies the layer name of a shape.

The data type of *layer* is string.

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

Specifies the layer name of a shape.

The data type of *layer_name* is string.

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

Specifies the layer number of a shape.

The data type of *layer_number* is integer.

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

Specifies the length of a wire or path shape in user units.

The data type of *length* is float.

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

Specifies the coloring information for a shape. This information is needed in the multiple-patterning flow.

The data type of *multiple_pattern_mask_constraint* is string.

Its valid values are

- *any_mask*
- *mask1_soft*
- *mask1_hard*
- *mask2_soft*
- *mask2_hard*
- *mask3_soft*
- *mask3_hard*
- *same_mask*

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

Specifies the name of a shape.

The data type of *name* is string.

This attribute is read-only.

Specifies the object ID of the net associated with a shape.

The data type of *net_id* is integer.

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

Specifies the type of net associated with a shape.

The data type of *net_type* is string.

This attribute is read-only.

Specifies the object class name of a shape, which is *shape*.

The data type of *object_class* is string.

This attribute is read-only.

Specifies the object ID in the Milkyway design file.

The data type of *object_id* is integer.

This attribute is read-only.

Specifies the object type, which can be *RECTANGLE*, *POLYGON*, *TRAPEZOID*, *PATH*, *HWIRE*, or *VWIRE*.

The data type of *object_type* is string.

The attribute is read-only.

Specifies the Milkyway design file name in which a shape is located.

The data type of *owner* is string.

This attribute is read-only.

Specifies the name of the net to which a shape is connected.

The data type of *owner_net* is string.

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

Feedback

Specifies the point list of a shape's boundary.

The data type of *points* is string.

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

Specifies the route type of a shape.

The data type of *route_type* is string.

Its valid values are

- *User Enter* or *user_enter*
- *Signal Route* or *signal_route*
- *Signal Route (Global)* or *signal_route_global*
- *P/G Ring* or *pg_ring*
- *Clk Ring* or *clk_ring*
- *P/G Strap* or *pg_strap*
- *Clk Strap* or *clk_strap*
- *P/G Macro/IO Pin Conn* or *pg_macro_io_pin_conn*
- *P/G Std. Cell Pin Conn* or *pg_std_cell_pin_conn*
- *Zero-Skew Route* or *clk_zero_skew_route*
- *Bus* or *bus*
- *Shield (fix)* or *shield*
- *Shield (dynamic)* or *shield_dynamic*
- *Fill Track* or *clk_fill_track*
- *Unknown* or *unknown*

The *mw_attr_value_no_space* variable determines whether the *get_attribute* or *report_attribute* command returns the value containing spaces or underscores.

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

Specifies the width of a wire or path shape in user units.

The data type of *width* is float.

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

See Also

- get_attribute
- list_attributes
- report_attribute
- set_attribute

---

site_row_attributes

Contains attributes related to site row.

Description

Contains attributes related to site row.

You can use *get_attribute* to determine value of an attribute, and use *report_attribute* to get a report of all attributes on specified site row. Specified with *list_attribute -class site_row -application*, the definition of attributes can be listed.

Site Row Attributes

Specifies Milkyway design ID in which a site row object is located.

This attribute is read-only.

Specifies name of a site row object.

This attribute is read-only.

Specifies object class name of a site row object, which is *site_row*.

This attribute is read-only.

Specifies object ID in Milkyway design file.

This attribute is read-only.

Specifies the bbox of a site row object.

This attribute is read-only.

Specifies the lower-left of a site row object.

This attribute is read-only.

Specifies x coordinate of the lower_left of the bbox a site row object.

This attribute is read-only.

Specifies y coordinate of the lower_left of the bbox a site row object.

This attribute is read-only.

Specifies the upper-right of the bbox of a site row object.

This attribute is read-only.

Specifies x coordinate of the upper-right of the bbox a site row object.

This attribute is read-only.

Specifies y coordinate of the upper-right of the bbox a site row object.

This attribute is read-only.

Specifies the type of site being defined.

This attribute is read-only.

Specifies the orientation of the sites. The value can be N, W, S, E, FN, FW, FS, FE.

This attribute is read-only.

Specifies the direction of the row. The value can be v, h, vertical and horizontal.

This attribute is read-only.

Specifies the number of the sites in the row.

This attribute is read-only.

Specifies the space for each site, from the lower left corner of the site to the lower left corner of the next site. The value is specified in microns.

This attribute is read-only.

Specifies the lower left corner of the row, regardless of orientation.

This attribute is read-only.

Specifies the restricted orientations of placed cells on a specified row, which are based on what direction of row is, whether row is flipped and what direction of unit tile is.

This attribute is read-only.

It is just a positive integer associated with site rows. Later on set_cell_row_type can be used to associate one particular cell with some particular rows of corresponding row_type.

This attribute can be read, set and removed.

TRUE: ignore specified site row FALSE: The specified site row is not ignored

Feedback

The attribute can be read and set.

The attribute is true if the site row has the property "is_reserved_placement_area".

The attribute is read-only.

See Also

- get_attribute
- list_attributes
- report_attribute

---

supply_net_attributes

Describes the attributes related to supply nets.

Description

This man page describes the attributes related to supply nets. These attributes are defined only in UPF mode.

You can use the *get_attribute* command to determine value of an attribute, and use the *report_attribute* command to get a report of all the attributes on a specified supply net. To see all supply net attributes, use the *list_attribute -class supply_net -application* command.

Supply Net Attributes

cell_id

Specifies Milkyway design ID in which a supply net object is located.

This attribute is read-only.

internal_supply_net

Specifies whether a supply net object is internal.

When the supply net is set to internal, it can be connected with switched macro internal PG pins, derive_pg_connection won't try to connect this internal supply net.

This attribute is writable. You can use *set_attribute* to set its value.

name

Specifies name of a supply net object.

This attribute is read-only.

Feedback

full_name

Specifies full name of a supply net object.

This attribute is read-only.

object_class

Specifies object class name of a supply net object, which is *supply_net*.

This attribute is read-only.

object_id

Specifies object ID in Milkyway design file.

This attribute is read-only.

within_block_abstraction

Specifies whether the supply net is part of the block abstraction.

The data type of *within_block_abstraction* is boolean.

This attribute is read-only and cannot be modified.

within_ilm

Specifies whether the supply net is part of an ILM.

The data type of *within_ilm* is boolean.

This attribute is read-only and cannot be modified.

See Also

- get_attribute
- list_attributes
- report_attribute

---

supply_port_attributes

Describes the attributes related to supply ports.

Description

This man pages describes the attributes related to supply ports. The attributes are defined only in UPF mode.

You can use the *get_attribute* command to determine value of an attribute, and use the *report_attribute* command to get a report of all attributes on a specified supply port. To see all supply port attributes, use the *list_attribute -class supply_port -application* command.

Supply Port Attributes

cell_id

Specifies Milkyway design ID in which a supply port object is located.

This attribute is read-only.

direction

Specifies the direction of a supply port object. The value can be "in" or "out".

This attribute is read-only.

full_name

Specifies full name of a supply port object.

This attribute is read-only.

name

Specifies name of a supply port object.

This attribute is read-only.

object_class

Specifies object class name of a supply port object, which is *supply_port*.

This attribute is read-only.

object_id

Specifies object ID in Milkyway design file.

This attribute is read-only.

within_block_abstraction

Specifies whether the supply port is part of the block abstraction.

The data type of *within_block_abstraction* is Boolean.

This attribute is read-only and cannot be modified.

within_ilm

Specifies whether the supply port is part of an ILM.

The data type of *within_ilm* is Boolean.

Feedback

This attribute is read-only and cannot be modified.

See Also

- get_attribute
- list_attributes
- report_attribute

t

terminal_attributes

Description

This man page described the attributes related to terminals (physical pins).

You can use the *get_attribute* command to determine value of an attribute, and use the *report_attribute* command to get a report of all the attributes on a specified object. To see all terminal attributes, use the *list_attribute -class terminal -application* command.

Note that attributes on a diode terminal are read-only.

Terminal Attributes

access_direction

Specifies allowable access directions for a terminal object.

The data type of *access_direction* is string.

Its valid values are:

- **right**
- **left**
- **up**
- **down**
- **unknown**

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Note that this attribute is undefined on diode terminal. You will get warning message with message ID "ATTR-3" when you try to get its value on diode terminal.

bbox

Specifies the bounding-box of a terminal. The *bbox* is represented by a *rectangle*.

The format of a *rectangle* specification is {{llx lly} {urx ury}}, which specifies the lower-left and upper-right corners of the rectangle.

The data type of *bbox* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

bbox_ll

Specifies the lower-left corner of the bounding-box of a terminal.

The *bbox_ll* is represented by a *point*. The format of a *point* specification is {x y}.

You can get the *attr_name* of a terminal, by accessing the first element of its *bbox*.

The data type of *bbox_ll* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

bbox_llx

Specifies x coordinate of the lower-left corner of the bounding-box of a terminal.

The data type of *bbox_llx* is double.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

bbox_lly

Specifies y coordinate of the lower-left corner of the bounding-box of a terminal.

The data type of *bbox_lly* is double.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Feedback

bbox_ur

Specifies the upper-right corner of the bounding-box of a terminal.

The box_ur is represented by a *point*. The format of a *point* specification is {x y}.

You can get the *bbox_ur* of a terminal, by accessing the second element of its *bbox*.

The data type of *bbox_ur* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

bbox_urx

Specifies x coordinate of the upper-right corner of the bounding-box of a terminal.

The data type of *bbox_urx* is double.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

bbox_ury

Specifies y coordinate of the upper-right corner of the bounding-box of a terminal.

The data type of *bbox_ury* is double.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

cell_id

Specifies Milkyway design ID in which a terminal object is located.

The data type of *cell_id* is integer.

This attribute is read-only.

constrained_status

Specifies constrained status of a terminal object.

The data type of *constrained_status* is string.

Its valid values are:

- **`manual_created`**
- **`side`**

- **location**
- **order**
- **unknown**

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Note that this attribute is undefined on diode terminal. You will get warning message with message ID "ATTR-3" when you try to get its value on diode terminal.

direction

Specifies direction of a terminal object.

The data type of *direction* is string.

Its valid values are:

- **in**
- **out**
- **inout**
- **tristate**
- **unknown**
- **input**
- **output**

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Note that the direction of a diode terminal will be marked as *diode*. You can not change its value by *set_attribute*.

double_pattern_mask_constraint

Specifies the coloring information of the terminal. This information is needed in the double patterning flow.

The data type of *double_pattern_mask_constraint* is string.

Its valid values are:

- **any_mask**
- **mask1_soft**
- **mask1_hard**
- **mask2_soft**
- **mask2_hard**
- **same_mask**

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

eeq_class

Specifies electrically equivalent class of a terminal object.

The data type of *eeq_class* is integer.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Note that this attribute is undefined on diode terminal. You will get warning message with message ID "ATTR-3" when you try to get its value on diode terminal.

is_diode

Indicates whether a terminal object is diode.

The data type of *is_diode* is boolean.

This attribute is read-only.

is_fixed

Specifies whether a terminal object is marked as fixed.

The data type of *is_fixed* is boolean.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

layer

Specifies layer name of a terminal object.

The data type of *layer* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

must_join_class

Specifies must-join class of a terminal.

The data type of *must_join_class* is integer.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Note that this attribute is undefined on diode terminal. You will get warning message with message ID "ATTR-3" when you try to get its value on diode terminal.

multiple_pattern_mask_constraint

Specifies the coloring information of the terminal. This information is needed in the multiple patterning flow.

The data type of *multiple_pattern_mask_constraint* is string.

Its valid values are:

- **any_mask**
- **mask1_soft**
- **mask1_hard**
- **mask2_soft**
- **mask2_hard**
- **mask3_soft**

- **mask3_hard**

- **same_mask**

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

name

Specifies name of a terminal object.

The data type of *name* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

number_of_points

Specifies the number of points to illustrate the boundary of a terminal object.

The data type of *number_of_points* is integer.

You can refer to the attribute *points*. The list length of *points* is the value of *number_of_points*.

This attribute is read-only.

object_class

Specifies object class name of a terminal, which is *terminal*.

The data type of *object_class* is string.

This attribute is read-only.

object_id

Specifies object ID in Milkyway design file.

The data type of *object_id* is integer.

This attribute is read-only.

owner

Specifies Milkyway design file name in which a terminal is located.

The data type of *owner* is string.

This attribute is read-only.

Feedback

owner_port

Specifies port name which a terminal object is associated with.

The data type of *owner_port* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

points

Specifies points of the boundary of a terminal. A terminal can be a rectangle, a rectilinear polygon, or multiple rectangles.

When a terminal is either a rectangle or a rectilinear polygon, its *points* is represented by a list of points. The last element of the list is the same as the first element.

When a terminal consists of multiple rectangles, its *points* is represented by a list of points of rectangles. Every five points represent one rectangle.

The data type of *points* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

side_status

Specifies constrained side name.

The data type of *side_status* is string.

Its valid values are:

- **left**
- **right**
- **bottom**
- **top**
- **unknown**

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Feedback

Note that this attribute is undefined on diode terminal. You will get warning message with message ID "ATTR-3" when you try to get its value on diode terminal.

status

Specifies the placement status of a terminal object.

The data type of *status* is string. The valid values are:

- **fixed**
- **placed**
- **cover**
- **unplaced**

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Note that this attribute is undefined on diode terminal. You will get an ATTR-3 warning message when you try to get its value on diode terminal.

See Also

- get_attribute
- list_attributes
- report_attribute
- set_attribute

---

text_attributes

Contains attributes related to text.

Description

Contains attributes related to text.

You can use *get_attribute* to determine value of an attribute, and use *report_attribute* to get a report of all attributes on specified object. Specified with *list_attribute -class text -application*, the definition of attributes can be listed.

Feedback

Text Attributes

Specifies the anchor position for text from its *origin*.

The data type of *anchor* is string.

Its valid values are:

- **lb** - Left Bottom
- **cb** - Center Bottom
- **rb** - Right Bottom
- **lc** - Left Center
- **c** - Center Center
- **rc** - Right Center
- **lt** - Left Top
- **ct** - Center Top
- **rt** - Right Top

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies the bounding-box of a text. The *bbox* is represented by a *rectangle*.

The format of a *rectangle* specification is {{llx lly} {urx ury}}, which specifies the lower-left and upper-right corners of the rectangle.

The data type of *bbox* is string.

This attribute is read-only.

Specifies the lower-left corner of the bounding-box of a text.

The *bbox_ll* is represented by a *point*. The format of a *point* specification is {x y}.

You can get the *attr_name* of a text, by accessing the first element of its *bbox*.

The data type of *bbox_ll* is string.

This attribute is read-only.

Specifies x coordinate of the lower-left corner of the bounding-box of a text.

The data type of *bbox_llx* is double.

This attribute is read-only.

Specifies y coordinate of the lower-left corner of the bounding-box of a text.

The data type of *bbox_lly* is double.

This attribute is read-only.

Specifies the upper-right corner of the bounding-box of a text.

The box_ur is represented by a *point*. The format of a *point* specification is {x y}.

You can get the *bbox_ur* of a text, by accessing the second element of its *bbox*.

The data type of *bbox_ur* is string.

This attribute is read-only.

Specifies x coordinate of the upper-right corner of the bounding-box of a text.

The data type of *bbox_urx* is double.

This attribute is read-only.

Specifies y coordinate of the upper-right corner of the bounding-box of a text.

The data type of *bbox_ury* is double.

This attribute is read-only.

Specifies Milkyway design ID in which a text object is located.

The data type of *cell_id* is integer.

This attribute is read-only.

Specifies height of a text object.

The data type of *height* is float.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies layer name of a text object.

The data type of *layer* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Feedback

Specifies name of a text object.

The data type of *name* is string.

This attribute is read-only.

Specifies object class name of a text, which is *text*.

The data type of *object_class* is string.

This attribute is read-only.

Specifies object ID in Milkyway design file.

The data type of *object_id* is integer.

This attribute is read-only.

Specifies orientation of a text object.

The data type of *orientation* is string.

Its valid values are:

- **N**
- **E**
- **S**
- **W**
- **FN**
- **FE**
- **FS**
- **FW**

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies origin of a text object.

The *origin* is represented by a *point*. The format of a *point* specification is {x y}.

Feedback

The data type of *origin* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

Specifies the text string to create and display.

The data type of *text* is string.

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

See Also

- get_attribute
- list_attributes
- report_attribute
- set_attribute

---

track_attributes

Contains attributes related to track.

Description

Contains attributes related to track.

You can use *get_attribute* to determine value of an attribute, and use *report_attribute* to get a report of all attributes on specified track. Specified with *list_attribute -class track -application*, the definition of attributes can be listed.

Track Attributes

Specifies the bounding-box of a track. The *bbox* is represented by a *rectangle*.

The format of a *rectangle* specification is {{llx lly} {urx ury}}, which specifies the lower-left and upper-right corners of the rectangle.

The *bbox* of a track is calculated by the *origin* and *orientation* of its cell and the actual *bbox* of its corresponding terminal from the child MW design.

This attribute is read-only.

Specifies the lower-left corner of the bounding-box of a track.

The *bbox_ll* is represented by a *point*. The format of a *point* specification is {x y}.

You can get the *bbox_ll* of a track by accessing the first element of its *bbox*.

This attribute is read-only.

Specifies x coordinate of the lower-left corner of the bounding-box of a track.

The data type of *bbox_llx* is double.

This attribute is read-only.

Specifies y coordinate of the lower-left corner of the bounding-box of a track.

The data type of *bbox_lly* is double.

This attribute is read-only.

Specifies the upper-right corner of the bounding-box of a track.

The *bbox_ur* is represented by a *point*. The format of a *point* specification is {x y}.

You can get the *bbox_ur* of a track by accessing the second element of its *bbox*.

This attribute is read-only.

Specifies x coordinate of the upper-right corner of the bounding-box of a track.

The data type of *bbox_urx* is double.

This attribute is read-only.

Specifies y coordinate of the upper-right corner of the bounding-box of a track.

The data type of *bbox_ury* is double.

This attribute is read-only.

Specifies Milkyway design ID in which a track object is located.

This attribute is read-only.

Specifies the number of grid of a track.

The data type of *count* is integer.

This attribute is read-only.

Specifies the routing direction of a track.

The valid values can be: X, Y.

This attribute is read-only.

Specifies the layer name where a track object is located on.

This attribute is read-only.

Specifies the object name of a track.

This attribute is read-only.

Specifies object class name of a track, which is *track*.

This attribute is read-only.

Specifies object ID in Milkyway design file.

This attribute is read-only.

Specifies if a track is reserved for specified width.

The data type of *reserved_for_width* is boolean.

This attribute is read-only.

Specifies track step of a track.

The data type of *space* is integer.

This attribute is read-only.

Specifies start position of a track.

The data type of *start* is coordinate point.

This attribute is read-only.

Specifies end position of a track.

The data type of *stop* is coordinate point.

This attribute is read-only.

Specifies the width of wires associated with the track.

The data type of *width* is double.

This attribute is read-only.

See Also

- get_attribute
- list_attributes
- report_attribute

Feedback

v

vhdlout_local_attributes

This variable is obsolete.

via_attributes

Contains attributes related to via.

Description

Contains attributes related to via.

You can use the *get_attribute* command to determine value of an attribute, and use the *report_attribute* command to get a report of all attributes on a specified object. List the definitions of attributes using *list_attribute -class via -application*,

Via Attributes

array_size

Specifies the array size of a via object. The format is {*col row*}.

The data type of *array_size* is string.

This attribute is read-only.

bbox

Specifies the bounding box of a via. The *bbox* is represented by a *rectangle*.

The format of a *rectangle* specification is {{llx lly} {urx ury}}, which specifies the lower-left and upper-right corners of the rectangle.

The data type of *bbox* is string.

This attribute is read-only.

bbox_ll

Specifies the lower-left corner of the bounding box of a via.

The *bbox_ll* is represented by a *point*. The format of a *point* specification is {x y}.

You can get the *bbox_ll* of a via by accessing the first element of its *bbox*.

The data type of *bbox_ll* is string.

This attribute is read-only.

bbox_llx

Specifies the x coordinate of the lower-left corner of the bounding box of a via.

The data type of *bbox_llx* is integer.

This attribute is read-only.

bbox_lly

Specifies the y coordinate of the lower-left corner of the bounding box of a via.

The data type of *bbox_lly* is integer.

This attribute is read-only.

bbox_ur

Specifies the upper-right corner of the bounding box of a via.

The *bbox_ur* is represented by a *point*. The format of a *point* specification is {x y}.

You can get the *bbox_ur* of a via, by accessing the second element of its *bbox*.

The data type of *bbox_ur* is string.

This attribute is read-only.

bbox_urx

Specifies the x coordinate of the upper-right corner of the bounding box of a via.

The data type of *bbox_urx* is integer.

This attribute is read-only.

bbox_ury

Specifies the y coordinate of the upper-right corner of the bounding box of a via.

The data type of *bbox_ury* is integer.

This attribute is read-only.

cell_id

Specifies the Milkyway design ID in which a via object is located.

The data type of *cell_id* is integer.

This attribute is read-only.

center

Specifies the center position of a via object.

The data type of *center* is string.

This attribute is read-only.

col

Specifies the number of horizontal columns of a via object.

The data type of *col* is integer.

This attribute is read-only.

is_via_ladder

Specifies the via ladder is true or not.

The data type of *is_via_ladder* is boolean.

This attribute is writable.

layer

Specifies the layer name list with which a via object is associated. The format is {*via_layer lower_layer upper_layer*}.

The data type of *layer* is string.

This attribute is read-only.

lower_layer

Specifies the lower layer name.

The data type of *lower_layer* is string.

This attribute is read-only.

lower_layer_double_pattern_mask_constraint

Specifies the coloring information for lower layer of the via. This information is needed in the double patterning flow.

The data type of *lower_layer_double_pattern_mask_constraint* is string.

The following values are valid:

- **`any_mask`**
- **`mask1_soft`**
- **`mask1_hard`**

- **mask2_soft**
- **mask2_hard**
- **same_mask**

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

lower_layer_multiple_pattern_mask_constraint

Specifies the coloring information for lower layer of the via. This information is needed in the multiple patterning flow.

The data type of *lower_layer_multiple_pattern_mask_constraint* is string.

The following values are valid:

- **any_mask**
- **mask1_soft**
- **mask1_hard**
- **mask2_soft**
- **mask2_hard**
- **mask3_soft**
- **mask3_hard**
- **same_mask**

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

name

Specifies the name of a via object.

The data type of *name* is string.

This attribute is read-only.

net_id

Specifies the object ID of the net associated with a via object.

The data type of *net_id* is integer.

This attribute is writable. Use the *set_attribute* command to modify its value on a specified object.

net_type

Specifies the type of net associated with a via object.

The data type of *net_type* is string.

This attribute is read-only.

object_class

Specifies the object class name of a via object, which is *via*.

The data type of *object_class* is string.

This attribute is read-only.

object_id

Specifies the object ID in a Milkyway design file.

The data type of *object_id* is integer.

This attribute is read-only.

object_type

Specifies the object type name, which can be *via*, *via_array*, or *via_cell*.

The data type of *object_type* is string.

This attribute is read-only.

orientation

Specifies the orientation of a via object.

The data type of *orientation* is string.

The following values are valid:

- **N**

- **E**

Feedback

- **S**
- **W**
- **FN**
- **FE**
- **FS**
- **FW**

This attribute is writable. Use the *set_attribute* command to modify its value on a specified object. However, when the object type is via_cell, you cannot change its *orientation*.

owner

Specifies the Milkyway design file name in which a via is located.

The data type of *owner* is string.

This attribute is read-only.

owner_net

Specifies the net name to which a via is connected.

The data type of *owner_net* is string.

This attribute is writable. Use the *set_attribute* command to modify its value on a specified object.

route_type

Specifies the route type of a via.

The data type of *route_type* is string.

The following values are valid:

- **User Enter** or **user_enter**
- **Signal Route** or **signal_route**
- **Signal Route (Global)** or **signal_route_global**

- **P/G Ring** or **pg_ring**
- **Clk Ring** or **clk_ring**
- **P/G Strap** or **pg_strap**
- **Clk Strap** or **clk_strap**
- **P/G Macro/IO Pin Conn** or **pg_macro_io_pin_conn**
- **P/G Std. Cell Pin Conn** or **pg_std_cell_pin_conn**
- **Zero-Skew Route** or **clk_zero_skew_route**
- **Bus** or **bus**
- **Shield (fix)** or **shield**
- **Shield (dynamic)** or **shield_dynamic**
- **Fill Track** or **clk_fill_track**
- **Unknown** or **unknown**

The Tcl variable *mw_attr_value_no_space* determines whether *get_attribute* or *report_attribute* returns route_type containing spaces or underscores.

This attribute is writable. Use the *set_attribute* command to modify its value on a specified object.

row

Specifies the number of vertical rows in a via object.

The data type of *row* is integer.

This attribute is read-only.

Feedback

upper_layer

Specifies the upper layer name.

The data type of *upper_layer* is string.

This attribute is read-only.

upper_layer_double_pattern_mask_constraint

Specifies the coloring information for upper layer of the via. This information is needed in the double patterning flow.

The data type of *upper_layer_double_pattern_mask_constraint* is string.

The following values are valid:

- **any_mask**
- **mask1_soft**
- **mask1_hard**
- **mask2_soft**
- **mask2_hard**
- **same_mask**

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

upper_layer_multiple_pattern_mask_constraint

Specifies the coloring information for upper layer of the via. This information is needed in the multiple patterning flow.

The data type of *upper_layer_multiple_pattern_mask_constraint* is string.

The following values are valid:

- **any_mask**
- **mask1_soft**
- **mask1_hard**

- **mask2_soft**
- **mask2_hard**
- **mask3_soft**
- **mask3_hard**
- **same_mask**

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

via_layer

Specifies the via layer name.

The data type of *via_layer* is string.

This attribute is writable. Use the *set_attribute* command to modify its value on a specified object.

via_layer_double_pattern_mask_constraint

Specifies the coloring information for via layer of the via. This information is needed in the double patterning flow.

The data type of *via_layer_double_pattern_mask_constraint* is string.

The following values are valid:

- **any_mask**
- **mask1_soft**
- **mask1_hard**
- **mask2_soft**
- **mask2_hard**
- **same_mask**

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

via_layer_multiple_pattern_mask_constraint

Specifies the coloring information for via layer of the via. This information is needed in the multiple patterning flow.

The data type of *via_layer_multiple_pattern_mask_constraint* is string.

The following values are valid:

- **any_mask**
- **mask1_soft**
- **mask1_hard**
- **mask2_soft**
- **mask2_hard**
- **mask3_soft**
- **mask3_hard**
- **same_mask**

This attribute is writable. You can use *set_attribute* to modify its value on a specified object.

via_master

Specifies the via master's name defined in the library's technology file; for via_cell, it is the library cell's name.

The data type of *via_master* is string.

This attribute is writable. Use the *set_attribute* command to modify its value on a specified object.

x_pitch

Specifies the center-to-center spacing of a via object in the horizontal direction.

The data type of *x_pitch* is float.

Feedback

This attribute is writable. Use the *set_attribute* command to modify its value on a specified object.

y_pitch

Specifies the center-to-center spacing of a via object in the vertical direction.

The data type of *y_pitch* is float.

This attribute is writable. Use the *set_attribute* command to modify its value on a specified object.

See Also

- get_attribute
- list_attributes
- report_attribute
- set_attribute

---

via_region_attributes

Contains attributes related to via region.

Description

Contains attributes related to via region.

You can use *get_attribute* to determine value of an attribute, and use *report_attribute* to get a report of all attributes on specified object. Specified with *list_attribute -class via_region -application*, the definition of attributes can be listed.

Via Region Attributes

Specifies the bounding-box of a via region. The *bbox* is represented by a *rectangle*.

The format of a *rectangle* specification is {{llx lly} {urx ury}}, which specifies the lower-left and upper-right corners of the rectangle.

The data type of *bbox* is string.

This attribute is read-only.

Specifies the lower-left corner of the bounding-box of a via region.

The *bbox_ll* is represented by a *point*. The format of a *point* specification is {x y}.

You can get the *attr_name* of a via region, by accessing the first element of its *bbox*.

The data type of *bbox_ll* is string.

This attribute is read-only.

Specifies x coordinate of the lower-left corner of the bounding-box of a via region.

The data type of *bbox_llx* is double.

This attribute is read-only.

Specifies y coordinate of the lower-left corner of the bounding-box of a via region.

The data type of *bbox_lly* is double.

This attribute is read-only.

Specifies the upper-right corner of the bounding-box of a via region.

The *bbox_ur* attribute is represented by a point. The format of a point specification is {x y}.

You can get the *bbox_ur* of a via region, by accessing the second element of its *bbox*.

The data type of *bbox_ur* is string.

This attribute is read-only.

Specifies x coordinate of the upper-right corner of the bounding-box of a via region.

The data type of *bbox_urx* is double.

This attribute is read-only.

Specifies y coordinate of the upper-right corner of the bounding-box of a via region.

The data type of *bbox_ury* is double.

This attribute is read-only.

Specifies Milkyway design ID in which a via region object is located.

The data type of *cell_id* is integer.

This attribute is read-only.

Specifies name of a via region object.

The data type of *name* is string.

This attribute is read-only.

Specifies the number of points to illustrate the boundary of a via region object.

The data type of *number_of_points* is integer.

You can refer to the attribute *points*. The list length of *points* is the value of *number_of_points*.

This attribute is read-only.

Specifies object class name of a via region, which is *via_region*.

The data type of *object_class* is string.

This attribute is read-only.

Specifies object ID in Milkyway design file.

The data type of *object_id* is integer.

This attribute is read-only.

Specifies port name which a via region object is associated with.

The data type of *owner_port* is string.

This attribute is read-only.

Specifies points of the boundary of a via region. A via region can be a rectangle, a rectilinear polygon, or multiple rectangles.

When a via region is either a rectangle or a rectilinear polygon, its *points* is represented by a list of points. The last element of the list is the same as the first element.

When a via region consists of multiple rectangles, its *points* is represented by a list of points of rectangles. Every five points represent one rectangle.

The data type of *points* is string.

This attribute is read-only.

Specifies whether a via region is used for the contactCode object who is rotated 90 degree.

The data type of *is_rotate90* is boolean.

This attribute is read-only.

Specifies the name of the via master associated with a via region object.

The data type of *via_master* is string.

This attribute is read-only.

See Also

- get_attribute
- list_attributes
- report_attribute

Feedback

voltage_area_attributes

Description

This man page describes the attributes related to voltage areas.

You can use the *get_attribute* command to determine the value of an attribute and the *report_attribute* command to get a report of all the attributes on a specified object.

To see a list of all voltage area attributes, use the *list_attributes -application -class voltage_area* command.

Voltage Area Attributes

bbox

Specifies the bounding box of a voltage area.

The bounding box is represented by a rectangle. The format of a rectangle specification is {{*llx lly*} {*urx ury*}}, which specifies the lower-left and upper-right corners of the rectangle.

The data type of *bbox* is string.

This attribute is read-only.

bbox_ll

Specifies the lower-left corner of the bounding box of a voltage area.

The *bbox_ll* is represented by a point. The format of a point specification is {*x y*}.

You can get the *bbox_ll* of a voltage area by accessing the first element of its *bbox*.

The data type of *bbox_ll* is string.

This attribute is read-only.

bbox_llx

Specifies the x-coordinate of the lower-left corner of the bounding box of a voltage area.

The data type of *bbox_llx* is double.

This attribute is read-only.

bbox_lly

Specifies the y-coordinate of the lower-left corner of the bounding box of a voltage area.

The data type of *bbox_lly* is double.

This attribute is read-only.

bbox_ur

Specifies the upper-right corner of the bounding box of a voltage area.

The *box_ur* is represented by a point. The format of a point specification is {*x y*}.

You can get the *bbox_ur* of a voltage area by accessing the second element of its *bbox*.

The data type of *bbox_ur* is string.

This attribute is read-only.

bbox_urx

Specifies the x-coordinate of the upper-right corner of the bounding box of a voltage area.

The data type of *bbox_urx* is double.

This attribute is read-only.

bbox_ury

Specifies the y-coordinate of the upper-right corner of the bounding box of a voltage area.

The data type of *bbox_ury* is double.

This attribute is read-only.

cell_id

Specifies the identification number of the Milkyway design in which the voltage area object is located.

The data type of *cell_id* is integer.

This attribute is read-only.

color

Specifies the color for a voltage area and its leaf cells.

The data type of *color* is string.

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

guardband

Specifies the guardband spacing around a voltage area.

Its format is {*guardband_x guardband_y*}.

The guardband is the spacing along the boundary of a voltage area where cells cannot be placed because of the lack of power supply rails.

The data type of *guardband* is string.

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

guardband_x

Specifies the guardband width in the horizontal direction.

The data type of *guardband_x* is float.

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

guardband_y

Specifies the guardband width in the vertical direction.

The data type of *guardband_y* is float.

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

is_fixed

Specifies whether a voltage area is in a fixed location and therefore ignored during shaping.

The data type of *is_fixed* is Boolean.

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

max_buffer_distance

Specifies maximum net length that is allowed to be routed in the particular voltage area. The virtual router and Zroute global router honor it.

The data type of *max_buffer_distance* is float and the unit is microns.

This attribute is writable. You can use the *set_attribute* command to modify its value on a specified object.

modules

Specifies the collection of top node cells inside a voltage area.

The data type of *modules* is collection.

This attribute is read-only.

name

Specifies the name of a voltage area object.

The data type of *name* is string.

This attribute is read-only.

object_class

Specifies the object class name of a voltage area, which is *voltage_area*.

The data type of *object_class* is string.

This attribute is read-only.

object_id

Specifies the object identification number in the Milkyway design database.

The data type of *object_id* is integer.

This attribute is read-only.

points

Specifies the point list of the voltage area's boundary.

The data type of *points* is string.

This attribute is read-only.

utilization

Specifies the utilization of a voltage area, the fraction of the area that is occupied by cells.

The data type of *utilization* is float.

This attribute is read-only.

within_block_abstraction

Specifies whether the voltage area is part of a block abstraction model.

The data type of *within_block_abstraction* is Boolean.

This attribute is read-only and cannot be modified.

within_ilm

Specifies whether the voltage area is part of an interface logic model (ILM).

The data type of *within_ilm* is Boolean.

This attribute is read-only and cannot be modified.

See Also

- create_voltage_area
- get_attribute
- list_attributes
- report_attribute
- set_attribute