Comments?
Contact your Synopsys sales representative.
Man Page Viewing and Printing Instructions

The following sections describe how to set up your UNIX environment so you can view and print man pages.

Setting Up the UNIX Environment

Edit your .cshrc file to contain these lines:

```
setenv SYN_MAN_DIR synopsys_root/doc/syn/man
setenv MANPATH ${MANPATH}:${SYN_MAN_DIR}
```

SYN_MAN_DIR is a variable that contains the path to the man page directories, and synopsys_root represents the specific path to the Synopsys synthesis software directory at your site.

Viewing Man Pages from UNIX

Command

```
% man command_name
```

Variable

```
% man variable_name
```

Variable group

```
% man variable_group_name
```

All attributes

```
% man attributes
```

Attribute group

```
% man attribute_group_name
```

Error, warning, or information message

```
% aman message_id
```
Viewing Man Pages from dc_shell

Command
   dc_shell> man command_name

Variable
   dc_shell> man variable_name

Variable group
   dc_shell> man variable_group_name

All attributes
   dc_shell> man attributes

Attribute group
   dc_shell> man attribute_group_name

Error, warning, or information message
   dc_shell> man message_id

Printing Man Pages from UNIX

User command
   % lpr -t -P printer_name \ $SYN_MAN_DIR/fmt1/command_name.1

Synopsys command
   % lpr -t -P printer_name \ $SYN_MAN_DIR/fmt2/command_name.2

Variable
   % lpr -t -P printer_name \ $SYN_MAN_DIR/fmt3/variable_name.3

Variable group
   % lpr -t -P printer_name \ $SYN_MAN_DIR/fmt3/variable_group_name.3

All attributes
   % lpr -t -P printer_name \ $SYN_MAN_DIR/fmt3/attributes.3

Attribute group
   % lpr -t -P printer_name \ $SYN_MAN_DIR/fmt3/attribute_group_name.3

You cannot print error, warning, or information message man pages from UNIX.
**Printing Man Pages from dc_shell**
You cannot print man pages from dc_shell.

**User Commands**
Invoke user commands from a UNIX shell.

**bc_view**
Runs the BCView performance analysis tool.

```
bc_view [-f project_file]
[-license_preference package | individual]
```

**rtl_analyzer**
Runs RTL Analyzer.

```
rtl_analyzer [-f project_file]
[-license_preference package | individual]
```

**aman**
Displays Synopsys extended error messages.

```
aman [error_message_code]
```

**cache_ls**
Lists elements in a Synopsys cache.

```
cache_ls cache_dir reg_expr
```

**cache_rm**
Removes elements from a Synopsys cache.

```
cache_rm cache_dir reg_expr
```

**create_synlib_template**
Generates template synthetic library description files from analyzed HDL source.

```
int create_synlib_template
-root synopsys_root
[-format output_format]
[-dw_library dw_library_name]
[-sldb sldb_base_name]
[-subprogram file_name]
[-no_subprogram file_name -compile_script]
```
create_types

Extracts user-defined type information from VHDL package files.

```
create_types [-nc] [-w lib] [-v]
[-o logfile] file_list
```

db2sge

Transfers symbols and schematics from Design Analyzer/dc_shell to SGE.

```
db2sge -database file_name
[-add_search_path search_path_list]
[-bit_type type_name]
[-bit_vector_type type_name] [-bustaps]
[-designs design_list]
[-display_instance_names]
[-display_pin_names] [-display_symbol_names]
[-no_compound_names]
[-one_name logic_value_name]
[-output_lib directory_name] [-overwrite]
[-scale value]
[-tcf_package file_name]
[-unknown_name logic_value_name]
[-use_lib_section section_name]
[-use_packages package_list]
[-zero_name logic_value_name]
 |
-tech_lib library_name
[-add_search_path search_path_list]
[-display_pin_names]
[-display_symbol_names]
[-generic_lib library_name]
[-output_lib directory_name] [-overwrite]
[-sym_lib library_name]
```

dc_shell

Runs the Design Compiler command shell.

```
dc_shell [-f script_file]
[-x command_string] [-no_init]
[-checkout feature_list] [-tcl_mode]
[-timeout timeout_value] [-version]
[-behavioral] [-fpga]
[-syntax_check | -context_check]
```
design_analyzer
Runs the Design Analyzer menu interface in the X Window System.


dp_shell
Runs the DesignPower command shell.


fpga_analyzer
Runs an FPGA-specific version of the Design Analyzer menu interface in the X Window System.


fpga_shell
Runs an FPGA-specific version of the Design Analyzer command shell.

fpga_shell [-f script_file] [-x command_string] [-no_init] [-checkout feature_list] [-timeout timeout_value] [-version]

lc_shell
Runs the Library Compiler command shell.

lc_shell [-f script_file] [-x command_string] [-no_init] [-version]
library_compiler
  Runs the Library Compiler graphical interface in the X window system.

  library_compiler [-f script_file]
  [-x command_string]
  [-no_init] [-version]

ra_shell
  Runs the RTL Analyzer command shell.

  ra_shell [-f script_file][-x command_string]
  [-no_init]
  [-checkout feature_list]
  [-timeout timeout_value] [-version]

synenc
  Runs the Synopsys Encryptor for HDL source code.

  synenc [-r synopsys_root] file_list

synopsys_users
  Lists the current users of the Synopsys licensed features.

  synopsys_users [feature_list]
Synthesis Commands
This section contains the following subsections:

- Command Syntax
- Commands Specific to dcsh Mode
- Commands Specific to dctcl Mode

Command Syntax
Invoke these commands from within a synthesis tool. Unless otherwise noted, all commands are available in both dcsh and dctcl mode.

acs_check_directories
Checks ACS directory structure settings for correctness.

```
int acs_check_directories
```

acs_compile_design
Compiles the constrained RTL to a netlist using constraints propagated from the top-level design.

```
int acs_compile_design
[-destination pass_name]
[-prepare_only] [-force]
design_name
```

acs_create_directories
Creates the project directory tree for Automated Chip Synthesis.

```
int acs_create_directories
```
acs_get_path

Gets the path location for the specified file. To specify a file, you specify its file type and, for pass-dependent files, its pass directory. For use in dc_shell-t (Tcl mode of dc_shell) only.

string acs_get_path
-file_type filetype
[-mode read | write]
[-pass pass_name]
[[-name filename] [-append]]
[-relative]

acs_recompile_design

Compiles an unmapped constrained .db file using time budgets. The time budgets are created by using a previously mapped design.

int acs_recompile_design
-budget_source budget_pass
-destination destination_pass
[-source source_pass]
[-prepare_only] [-force]
design_name

acs_refine_design

Refines an already mapped design.

int acs_refine_design
[-source pass_name]
[-destination pass_name]
[-prepare_only] [-force]
design_name

acs_report_directories

Reports the current directory structure settings.

int acs_report_directories
[-file_types type_list]

add_module

Reads in a specified library file containing module functional information and uses it to update an existing technology library.

int add_module [-overwrite] [-permanent]
file_name library_name [-no_warnings]
add_to_collection
Adds objects to a collection, resulting in a new
collection. The base collection remains unchanged.
For use in dc_shell-t (Tcl mode of dc_shell) only.

```tcl
string add_to_collection collection object_spec [-unique]
```

after (dctcl-mode only)
Built-in Tcl command.

alias
Defines an alias for a command or lists current alias
definitions.

```tcl
int alias [identifier [expansion]]
```

all_clocks
Returns a list of all clocks in the current design.

```tcl
list all_clocks()
```

all_cluster_cells
Returns a list of cells contained in the specified
cluster.

```tcl
list all_cluster_cells [-hierarchy] cluster_name
```

all_clusters
Returns a list of subclusters associated with a
specified cluster or with the current design.

```tcl
list all_clusters [-cluster cluster_name] [-leaf]
```

all_connected
Returns the objects connected to a net, port, pin, or a
net or pin instance.

```tcl
list all_connected object
```
all_critical_cells
  Returns a list of critical leaf cells in the top hierarchy of the current design.

  list all_critical_cells
  [-slack_range range_value]

all_critical_pins
  Returns a list of critical endpoints or startpoints in the current design.

  list all_critical_pins
  [-type endpoint | -startpoint]
  [-slack_range range_value]

all_designs
  Returns a list of all designs in the current design.

  all_designs

all_inputs
  Returns a list of input or inout ports in the current design.

  list all_inputs
  [-clock clock_name]
  [-edge_triggered | -level_sensitive]

all_outputs
  Returns a list of output or inout ports in the current design.

  list all_outputs
  [-clock clock_name]
  [-edge_triggered | -level_sensitive]

all_registers
  Returns a list of sequential cells or pins in the current design.

  list all_registers [-no_hierarchy]
  [-clock clock_name] [-cells]
  [-data_pins] [-clock_pins]
  [-slave_clock_pins] [-output_pins]
  [-level_sensitive | -edge_triggered]
  [-master_slave]
allocate_budgets

Writes out a design database file and Tcl script, and invokes the budget_shell command allocate_budgets from within dc_shell; not valid in dc_shell-t.

```c
int allocate_budgets [-db database]
[-dont_start]
[-verbose] [-incremental] [-check_only]
[-create_context]
[-no_boundary_annotations]
[-effort normal | level1 | level2]
[-min_register_to_output minimum_budget]
[-min_input_to_output minimum_budget]
[-min_input_to_register minimum_budget]
[-no_environment]
[-interblock_logic]
[-file_format_spec format_spec]
[-format dcsh | ptsh]
[-separator name_separator]
[-levels budget_levels]
[-write_context]
[cell_list]
```

allocate_partition_budgets

Creates budgets for the compile partitions in the specified design.

```bash
allocate_partition_budgets
-source src_pass
-type pre|post
-destination dst_pass
[-absolute_paths]
[-budget_shell budget_shell_exec]
[-transcript_exec transcript_exec]
[-format dcsh | dctcl]
[-overconstrain overcstr]
[-effort allocation_effort]
[-interblock_logic]
[-min_register_to_output minimum_budget]
[-min_input_to_output minimum_budget]
[-min_input_to_register minimum_budget]
[-no_environment]
[-generate_script_only]
[-remove]
[design]
```
analyze

Analyzes HDL files and stores the intermediate format for the HDL description in the specified library.

```plaintext
int analyze
[-library library_name] [-work library_name]
[-format vhdl | verilog]
[-create_update] [-update]
[-define define_list]
file_list
```

annotate_activity

Sets (or resets) the toggle_rate and static_probability values for specified objects in the current design.

```plaintext
int annotate_activity
[-static_probability sp_value]
[-toggle_rate tr_value]
[-clock clock_port_name]
[-period period_value]
[-hier]
[-reset]
[-select ports|regs|all]
[-objects object_list]
[-instance instance_list]
```

append (dctcl-mode only)

Built-in Tcl command.

apropos (dctcl-mode only)

Built-in Tcl command.

array (dctcl-mode only)

Built-in Tcl command.

auto_execok (dctcl-mode only)

Built-in Tcl command.

auto_import (dctcl-mode only)

Built-in Tcl command.

auto_load (dctcl-mode only)

Built-in Tcl command.
**auto_load_index** (dctcl-mode only)
Built-in Tcl command.

**auto_qualify** (dctcl-mode only)
Built-in Tcl command.

**balance_buffer**
Builds a balanced buffer tree on user-specified nets and drivers.

```
int balance_buffer
[-verify] [-verify_hierarchically]
[-verify_effort low | medium | high]
[-from start_point_list]
[-to end_point_list]
[-net net_list] [-depth int_value]
```

**balance_registers**
Moves the registers of a design to achieve a minimum cycle time.

```
int balance_registers [design_name]
```

**bc_check_design**
Performs a quick check of Behavioral Compiler scheduling information on the current design and reports incorrect coding styles.

```
int bc_check_design
[-io_mode cycle_fixed | superstate_fixed]
[-constraints]
```

**bc_dont_register_input_port**
For Behavioral Compiler, disables automatic register allocation for the specified input port.

```
int bc_dont_register_input_port port_name
```
**bc_dont_ungroup**
Prevents the compile command from ungrouping cells grouped by Behavioral Compiler, for the specified group classes or for all grouped cells in the current design.

```
int bc_dont_ungroup [-register] [-fsm]
[-random_logic] [-prio_logic]
[-array_logic]
```

**bc_group_process**
For Behavioral Compiler, creates one level of hierarchy instead of flattening each process after scheduling.

```
int bc_group_process [-with_memory]
```

**bc_margin**
Sets the timing margin used by Behavioral Compiler.

```
int bc_margin [-process process_name]
[-global margin]
[-reg margin] [-fsm margin ] [-mux margin]
[-preferred_FF cell_name] [-report_FF]
```

**bc_report_arrays**
Reports conflicting and non-conflicting accesses to arrays mapped to register files within an elaborated behavioral design.

```
bc_report_arrays [-conflicting]
[-non_conflicting]
```

**bc_report_memories**
Reports specific information about the memories instantiated within an elaborated behavioral design and also memories in the available synthetic libraries.

```
bca_report_memories [-synthetic_libraries]
[-bindings]
[-used_memories] [-conflicting]
[-non_conflicting]
```
**bc_time_design**
Calculates timing and area estimates and annotates them on the current design for Behavioral Compiler.

```
int bc_time_design
[-force] [-fastest]
[-cache_preserved_functions library_name]
[-except designs]]
[-use_cached_preserved_functions]
library_name
[-recompile designs]]
```

**bc_view**
Invokes BCView on the current design.

```
int bc_view [-output out_db_file]
[-project_file project_file_name]
[-dont_start]
[-cossap] [-search_additional path_list]
[-host machine_name]
[-arch architecture_name]
```

**begin_incr_mode**
Loads the current design into the Synopsys optimization engine and initiates the incremental mode. For use in dc_shell -tcl_mode only.

```
int begin_incr_mode [-load_annotation]
[-verify] [-verify_hierarchically]
[-verify_effort low | medium | high]
```

**binary** (dctcl-mode only)
Built-in Tcl command.

**break**
Immediately exits a loop structure.

```
int break
```

**calculate rtl load**
Calculates rtl load values based on layout-based annotation.

```
int calculate_rtl_load [-capacitance]
[-delay] pin_net_list
```
**catch** (dctcl-mode only)

Built-in Tcl command.

**cd**

Changes the current directory.

```tcl
int cd [directory]
```

**cell_of**

Returns the cell objects for given pins in the current design.

```tcl
list cell_of [object_list]
```

**chain_operations**

Specifies a list of operations to be scheduled by Behavioral Compiler for a specified process or for all processes.

```tcl
int chain_operations [-process process_name] operation_names
```

**change_link**

Changes the design to which a cell is linked.

```tcl
int change_link object_list design_name
```

**change_names**

Changes the names of ports, cells, and nets in a design.

```tcl
int change_names [-rules name_rules]
[-hierarchy] [-verbose]
[-names_file names_file]
```

**characterize**

Captures information about the environment of specific cell instances and assigns the information as attributes on the design to which the cells are linked.

```tcl
int characterize cell_list [-no_timing]
[-constraints]
[-connections] [-power] [-verbose]```
check_bindings
Checks the bindings in a synthetic library module definition.

int check_bindings [-bindings binding_list] [-pin_widths pin_width_list] module_name

check_bsd
Checks whether a design's boundary-scan implementation is compliant with IEEE Std 1149.1.

int check_bsd [-verbose] [-effort low | medium | high]

check_design
Checks the current design for consistency.

int check_design [-summary] [-no_warnings] [-one_level] [-post_layout | -only_post_layout]

check_dft
Checks a design against the design rules of a scan test methodology, with test points that have been inserted by either Autofix or Shadow LogicDFT.

int check_dft [-verbose] [-check_contention true | false | scan_shift_only | capture_only] [-check_float true | false | scan_shift_only | capture_only]

check_error
Prints extended information on errors from last command.

int check_error [-verbose] [-reset]

check_implementations
Checks the implementations in a synthetic library module definition.

int check_implementations [-implementations implementation_list] [-parameters parameter_list] module_name
check_scan
Checks a design against the design rules of a scan test methodology.

```
int check_scan [-verbose]
[-check_contention true | false]
[-check_float true | false]
```

check_synlib
Performs semantic checks on synthetic libraries.

```
int check_synlib
```

check_test
Checks a design against the design rules of a scan test methodology.

```
int check_test [-verbose]
[-check_contention true | false |
scan_shift_only | capture_only]
[-check_float true | false | scan_shift_only |
capture_only]
```

check_timing
 Warns about possible timing problems in the current design.

```
int check_timing
[-overlap_tolerance minimum_distance]
```

check_unmapped
Checks for any unmapped design below the current design.

```
int check_unmapped
```

clean_buffer_tree
Removes the buffer tree at a given driver pin on a mapped design.

```
int clean_buffer_tree
[-from start_point_list |
-to end_point_list | -net net_list]
[-hierarchy]
```

clock (dctcl-mode only)
Built-in Tcl command.
close (dctcl-mode only)
    Built-in Tcl command.

compare_collections
    Compares the contents of two collections. If the same objects are in both collections, the result is 0 (like string compare). If they are different, the result is nonzero. The order of the objects can optionally be considered. For use in dc_shell-t (Tcl mode of dc_shell) only.

    int compare_collections [-order_dependent] collection1 collection2

compare_design
    Compares two designs for functional equivalence.

    int compare_design [-effort low | medium | high] [-jtag] [-verbose] [-hierarchical] design1 design2

compare_fsm
    Compares the sequential behavior of two finite state machine designs.

    int compare_fsm design1 design2

compare_lib
    Performs a cross-reference check between a technology library and a symbol library.

    int compare_lib library1 library2
**compile**

Performs logic-level and gate-level synthesis and optimization on the current design.

```bash
int compile
[-no_map] [-map_effort low | medium | high]
[-area_effort none | low | medium | high]
[-incremental_mapping] [-exact_map]
[-verify] [-verify_hierarchically]
[-verify_effort low | medium | high]
[-ungroup_all] [-boundary_optimization]
[-no_design_rule | -only_design_rule]
[-scan] [-background run_name]
[-host machine_name]
[-arch architecture]
[-xterm] [-top]
```

**compile_partitions**

Distributes compile jobs for a design.

```bash
compile_partitions
-destination pass
-jobs njobs
[-make make_cmd]
```

**compile_preserved_functions**

Compiles and/or writes netlists for preserved functions.

```bash
int compile_preserved_functions
[ preserved_functions]
[-exclude preserved_functions]
[-no_compile]
[-compile_effort low | medium | high | 1 | 2 | 3]
[-include_script script_name]
[-write]
[-filename filename]
[-design_library library_name]
[-force_recompile]
[-stages number_of_stages]
[-clock_port_name clock_port_name]
[-output_delay delay_value]
[-input_delay delay_value]
[-sync_reset reset_port_name]
[-async_reset reset_port_name]
[-reset_polarity high | low]
```
**compile_systemc**

Reads a SystemC source file, checks for compliance with SystemC Compiler synthesis policy and syntax, and creates an internal database (.db) in the SystemC Compiler memory if there are no errors.

```plaintext
int compile_systemc [-cpp cpp_program] [-cpp_options options] file_name
```

**concat** (dtcl-mode only)

Built-in Tcl command.

**connect_net**

Connects a net to pins or ports.

```plaintext
int connect_net net object_list
```

**context_check**

Enables or disables the Syntax Checker context_check mode in which commands issued are checked for context errors.

```plaintext
int context_check true | false
```

**continue**

Begins the next loop iteration.

```plaintext
int continue
```

**copy_collection**

Duplicates the contents of a collection, resulting in a new collection. The base collection remains unchanged. For use in dc_shell-t (Tcl mode of dc_shell) only.

```plaintext
string copy_collection collection
```

**copy_design**

Copies a design to a new design, or copies a list of designs to a new file in dc_shell memory.

```plaintext
int copy_design source_design_name target_design_name
```
create_bsd
Creates ANSI/IEEE Std 1149.1-compliant boundary-scan circuitry using DesignWare macro cells.

    int create_bsd

create_bsd_patterns
Generates a set of functional patterns for a boundary-scan design.

    int create_bsd_patterns
    [-output test_program_name]
    [-effort low | medium | high]
    [-type vector_type_list]

create_bus
Creates a port bus or a net bus.

    int create_bus object_list bus_name ]
    [-type type_name]
    [-sort] [-no_sort]
    [-start start_bit] [-end end_bit]

create_cache
Populates the cache directories with instances of the requested synthetic modules.

    int create_cache -module module_list
    [-implementation implementation_list]
    [-parameters parameter_list]
    [-operating_condition operating_condition]
    [-wire_load list] [-report]

create_cell
Creates cells in the current design.

    int create_cell cell_list [reference_name]
    [-logic logic_value]

create_clock
Creates a clock object and defines its waveform in the current design.

    int create_clock [port_pin_list]
    [-name clock_name]
    [-period period_value] [-waveform edge_list]
**create_cluster**  
Creates a cluster in the physical hierarchy of the design.

```
int create_cluster [-name cluster_name] [-keep] [-multibits] [-parent parent_cluster_object] object_list
```

**create_command_group** (dctcl-mode only)  
Built-in Tcl command.

**create_design**  
Creates a design in dc_shell memory.

```
int create_design design_name [file_name]
```

**create_generated_clock**  
Creates a generated clock object.

```
string create_generated_clock [-name clock_name] [-source master_pin] [-divide_by divide_factor | -multiply_by multiply_factor] [-duty_cycle percent] [-invert] [-edges edge_list] [-edge_shift edge_shift_list] port_pin_list
```

**create_multibit**  
Creates a multibit component for the specified list of cells in the current design.

```
int create_multibit object_list [-name multibit_name] [-sort] [-no_sort]
```

**create_net**  
Creates nets in the current design.

```
int create_net net_list
```
create_operating_conditions

Creates a new set of operating conditions in a library.

    int create_operating_conditions
    -name name -library library_name
    -process process_value
    -temperature temperature_value
    -voltage voltage_value
    [-tree_type tree_type]
    [-calc_mode calc_mode]
    [-rail_volatages rail_value_pairs]

create_pass_directories

Creates the directory structure required for storing ACS data. For use in Tcl mode of dc_shell only.

    int create_pass_directories pass_list

create_port

Creates ports in the current design.

    int create_port port_list [-direction dir]

create_schematic

Generates a schematic for the current design.

    int create_schematic [-hierarchy]
    [-size sheet_size] [-portrait]
    [-fill_percent fill_value]
    [-outputs_attract]
    [-order_outputs output_port_list]
    [-sort_outputs] [-dont_left_justify_inputs]
    [-schematic_view] [-symbol_view]
    [-hier_view]
    [-no_bus] [-bit_mappers] [-implicit]
    [-no_rippers] [-sge]
    [-no_type_mappers] [-reference]
    [-gen_database]

create_test_clock

Defines the timing of a clock applied to a design during manufacturing test.

    int create_test_clock port_list
    -waveform two_value_rise_fall_edge_list
    [-period period_value]
    [-internal_clocks true | false | default]
create_test_patterns
Generates a set of test patterns for a design.

```c
int create_test_patterns
 [-input vector_file_name]
 [-output test_program_name]
 [-backtrack_effort low | medium | high]
 [-check_contention true | false]
 [-check_float true | false]
 [-compaction_effort low | medium | high | no_compaction]
 [-dft]
 [-max_cpu_per_fault CPU_time_per_fault]
 [-max_total_cpu total_CPU_time]
 [-max_random_patterns pattern_maximum]
 [-random_pattern_failure_limit failure_limit]
 [-sample percent] [-background run_name]
 [-host machine_name] [-arch architecture]
 [-xterm]
```

current_design
Sets the working design in dc_shell.

```c
string current_design [design]
```

current_design_name (dctcl-mode only)
Built-in Tcl command.

create_wire_load
Creates wire load models for the current design.

```c
int create_wire_load [-design design_name]
 [-cell cell_list] [-cluster cluster_name]
 [-hierarchy] [-this_level_nets_only]
 [-mode top | enclosed]
 [-name model_name]
 [-output file_name]
 [-update_lib library_name]
 [-write_script script_file_name]
 [-dont_smooth] [-trim trim_value]
 [-percentile percentile_value]
 [-total_area area]
 [-statistics]
```
current_instance
Sets the working instance object in dc_shell and enables other commands to be used on a specific cell in the design hierarchy.

```
string current_instance [instance]
```

date (dctcl-mode only)
Built-in Tcl command.

define_design_lib
Maps a design library to a UNIX directory.

```
int define_design_lib library_name -path directory
```

define_name_rules
Defines a set of name rules for designs.

```
```

define_proc_attributes (dctcl-mode only)
Built-in Tcl command.

delete_test
Deletes the current test program in dc_shell.

```
string delete_test
```

derive_clocks
Creates clock objects for network source pins or ports in the current design.

```
int derive_clocks
```
derive_constraints
Propagates design constraints and attribute settings from the top-level design to the specified subdesigns.

```
derive_constraints
[-attributes_only]
[-verbose]
[-budget]
cell_list
```

derive_timing_constraints
Derives timing requirements and places that constraint information on the current design.

```
int derive_timing_constraints
[-fix_hold] [-min_delay]
[-no_max_delay] [-no_max_period]
[-max_delay_scale max_delay_scale_factor]
[-min_delay_scale min_delay_scale_factor]
[-period_scale max_period_scale_factor]
[-separate_rise_and_fall]
```

disconnect_net
Disconnects a net from pins or ports.

```
int disconnect_net net object_list -all
```

dont_chain_operations
Constrains what operations are allowed to be chained by Behavioral Compiler for a specified process or for all processes.

```
int dont_chain_operations
[-process process_name] [-into]
[-from] {operation_names}
```

drive_of
Returns the drive resistance value of the specified library cell pin.

```
float drive_of library_cell_pin
[-rise | -fall] [-wire_drive]
[-piece best | worst | average | int_value]
```
**echo**
Displays literal strings and values of variables and expressions, in dc_shell or in dc_shell-t (Tcl mode of dc_shell).

**dc_shell:**
\[
\text{int} \ echo \ [-n] \ [\text{text}\_\text{string}] \ [\text{variable}] \ [(\text{expression})]
\]

**dc_shell-t:**
\[
\text{int} \ echo \ [-n] \ [\text{text}\_\text{string}] \ [$\text{variable}] \ [[\text{expr} \ \text{expression}]]
\]

**eco_align_design**
Aligns a pair of designs for use by the ECO Compiler product.
\[
\text{int} \ eco\_align\_design \ [-\text{out} \ \text{out}\_\text{file}]
\]

**eco_analyze_design**
Analyzes a pair of designs for structural correspondence.
\[
\text{int} \ eco\_analyze\_design
\]

**eco_current_design_pair**
Sets the working ECO design pair in dc_shell.
\[
\text{int} \ eco\_current\_design\_pair \ [-\text{design}\_\text{type} \ \text{design}]
\]

**eco_implement**
Computes the ECO implementation by modifying the new netlist.
\[
\text{int} \ eco\_implement \ [-\text{post}_\text{tapeout}]
\]

**eco_netlist_diff**
Prints the structural differences between the old netlist and the ECO netlist.
\[
\text{int} \ eco\_netlist\_diff
\]
**eco_recycle**
Recycles obsolete or spare cells in order to replace new cells generated by eco_implement.

```
int eco_recycle [-post_tapeout]
[-respect_physical_hier]
[-respect_distance distance_value]
```

**eco_report_cell**
Displays area and reference information about cells in the current instance, if set; or in the current design otherwise.

```
int eco_report_cell [-verbose]
```

**eco_reset_directives**
Resets the existing ECO Compiler directives.

```
int eco_reset_directives [-all] [-single]
```

**elaborate**
Builds a design from the intermediate format of a Verilog module, a VHDL entity and architecture, or a VHDL configuration.

```
int elaborate design_name
[-library library_name | -work library_name]
[-architecture arch_name]
[-parameters param_list]
[-file_parameters file_list] [-update]
[-schedule] [-gate_clock]
```

**encoding** *(dctcl-mode only)*
Built-in Tcl command.

**encrypt_lib**
Encrypts a VHDL source library file.

```
int encrypt_lib file_name
[-output encrypted_file]
```

**eof** *(dctcl-mode only)*
Built-in Tcl command.
**error** (dctcl-mode only)
   Built-in Tcl command.

**error_info** (dctcl-mode only)
   Built-in Tcl command.

**eval** (dctcl-mode only)
   Built-in Tcl command.

**exec** (dctcl-mode only)
   Built-in Tcl command.

**execute**
   Executes command arguments.
   ```
   int execute [-s] command [arguments]
   ```

**exit**
   Exits dc_shell.
   ```
   int exit [exit_code_value]
   ```

**expr** (dctcl-mode only)
   Built-in Tcl command.

**externalize_cell**
   Makes a cell instance used by Behavioral Compiler external to the current design after scheduling.
   ```
   int externalize_cell {cell_names}
   ```

**extract**
   Extracts a state-machine representation from a netlist or HDL description.
   ```
   int extract [-minimize] [-reachable]
   ```

**fblocked** (dctcl-mode only)
   Built-in Tcl command.

**fconfigure** (dctcl-mode only)
   Built-in Tcl command.
**fcopy** (dctcl-mode only)
Built-in Tcl command.

**file** (dctcl-mode only)
Built-in Tcl command.

**fileevent** (dctcl-mode only)
Built-in Tcl command.

**filter**
Returns a list of design objects that satisfy a conditional attribute expression.

list \texttt{filter} object\_list expression
[-\texttt{dont\_check\_real\_objects}]

**filter\_collection**
Filters a collection, resulting in a new collection. The base collection remains unchanged. For use in dc\_shell-t (Tcl mode of dc\_shell) only.

string \texttt{filter\_collection} collection expression
[-regexp] [-nocase]

**find**
Finds a design or library object.

list \texttt{find} type [name\_list] [-hierarchy] [-flat]

**flush** (dctcl-mode only)
Built-in Tcl command.

**for** (dctcl-mode only)
Built-in Tcl command.
foreach
Specifies the control structure for list traversal loop execution.

```
int foreach (variable_name, list_expression)
{
  loop_statement_block
}
```

foreach_in_collection
Iterates over the elements of a collection. For use in dc_shell-t (Tcl mode of dc_shell) only.

```
string foreach_in_collection itr_var
collections body
```

format (dctcl-mode only)
Built-in Tcl command.

get_attribute
Returns the value of an attribute on a list of design or library objects.

```
list get_attribute object_list
attribute_name [-bus] [-quiet]
```

get_cells
Creates a collection of cells from the current design relative to the current instance. You can assign these cells to a variable or pass them into another command. For use in Tcl-based dc_shell only.

```
string get_cells [-hierarchical]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-filter expression]
patterns | -of_objects objects
```
**get_clocks**

Creates a collection of clocks from the current design. You can assign these clocks to a variable or pass them into another command. For use in Tcl-based dc_shell only.

```tcl
```

**get_clusters**

Creates a collection of one or more clusters loaded into dc_shell. You can assign these clusters to a variable or pass them into another command. For use in Tcl-based dc_shell only.

```tcl
```

**get_design_lib_path**

Returns the directory to which the specified library is mapped.

```tcl
int get_design_lib_path library_name
```

**get_design_parameter**

Returns the value of a parameter on a parameterized design object. The parameter can be a generic in VHDL or a parameter in Verilog.

```tcl
int get_design_parameter parameter_name [-quiet]
```
get_designs
Creates a collection of one or more designs loaded into dc_shell. You can assign these designs to a variable or pass them into another command. For use in Tcl-based dc_shell only.

```
```

get_generated_clocks
Creates a collection of generated clocks.

```
```

get_lib_cells
Creates a collection of library cells from libraries loaded into dc_shell. You can assign these library cells to a variable or pass them into another command. For use in Tcl-based dc_shell only.

```
```

get_lib_pins
Creates a collection of library cell pins from libraries loaded into dc_shell. You can assign these library cell pins to a variable or pass them into another command. For use in Tcl-based dc_shell only.

```
```
get_libs
 Creates a collection of libraries loaded into dc_shell. You can assign these libraries to a variable or pass them into another command. For use in Tcl-based dc_shell only.

```
patterns | -of_objects objects
```

get_license
 Obtains a license for a feature.

```
int get_license feature_list
```

get_multibits
 Creates a collection of one or more multibits loaded into dc_shell. You can assign these multibits to a variable or pass them into another command. For use in dc_shell -tcl mode only.

```
```

get_nets
 Creates a collection of nets from the current design. You can assign these nets to a variable or pass them into another command. For use in Tcl-based dc_shell only.

```
patterns | -of_objects objects
```
get_object_name
   Returns the name of the object in a single-object
collection. For use in Tcl-based dc_shell only.

   string get_object_name collection

get_path_groups
   Creates a collection of path groups from the current
design. You can assign these path groups to a variable
or pass them into another command.

   string get_path_groups [-quiet]
   [-regexp]
   [-nocase]
   [-filter expression]
   patterns

get_pins
   Creates a collection of pins from the current design.
   You can assign these pins to a variable or pass them
   into another command. For use in Tcl-based dc_shell
   only.

   string get_pins [-hierarchical]
   [-filter expression]
   [-quiet]
   [-regexp]
   [-nocase]
   [-exact]
   [-leaf]
   patterns | of_objects objects

get_ports
   Creates a collection of ports from the current design.
   You can assign these ports to a variable or pass them
   into another command. For use in Tcl-based dc_shell
   only.

   string get_ports [-filter expression]
   [-quiet]
   [-regexp]
   [-nocase]
   [-exact]
   patterns | of_objects objects
get_references
Creates a collection of one or more references loaded into dc_shell. You can assign these references to a variable or pass them into another command. For use in dc_shell -tcl mode only.

string get_references [-hierarchical]
[-quiet]
[-regexp]
[-nocase]
[-exact]
[-filter expression]
patterns

get_timing_paths
Creates a collection of timing paths for custom reporting and other processing. You can assign these timing paths to a variable or pass them into another command.

string get_timing_paths [-to to_list]
[-from from_list]
[-through through_list]
[-delay_type delay_type]
[-nworst paths_per_endpoint]
[-max_paths max_path_count]
[-enable_preset_clear_arcs]
[-group group_name]
[-true]
[-true_threshold path_delay]
[-greater greater_limit]
[-lesser lesser_limit]
[-slack_greater_than greater_slack_limit]
[-slack_lesser_than lesser_slack_limit]

get_unix_variable
Returns the value of a UNIX environment variable.

string get_unix_variable variable_name

getenv (dctcl-mode only)
Built-in Tcl command.

gets (dctcl-mode only)
Built-in Tcl command.
**glob** (dctcl-mode only)
Built-in Tcl command.

**global** (dctcl-mode only)
Built-in Tcl command.

**group**
Creates a new level of hierarchy.

```
int group
    [cell_list | -logic | -pla | -fsm] [-soft
    | -hdl_block block_name | -hdl_all_blocks |
    -hdl_bussed]
    [-design_name design_name]
    [-cell_name cell_name]
    [-except exclude_list]
```

**group_path**
Groups a set of paths for cost function calculations.

```
int group_path [-weight weight_value]
    [-critical_range range_value]
    -default | -name group_name
    [-from from_list] [-to to_list]
    [-through through_list]
```

**group_variable**
Adds a variable to the specified variable group. This command is typically used by the system administrator only.

```
int group_variable group_name variable_name
```

**help**
In standard dc_shell (dcsh mode), the help command displays man pages for Synopsys commands. In Tcl-based dc_shell (dctcl mode), it displays quick help for one or more commands.

**dc_shell:**

```
int help [topic]
```

**dc_shell-t:**

```
string help [-verbose] pattern
```
highlight_path

Highlights timing paths in a schematic.

```
int highlight_path [pin_port_list]
[-from source_pins_or_ports]
[layer_name] [-critical_path] [-min] [-max]
[-min_rise] [-min_fall] [-max_rise]
[-max_fall]
[-no_auto_cycle]
```

history

Displays the history list (an ordered list of previously executed commands).

```
de_shell:
int history [-h] [-r] [n]

de_shell-t:
string history [-h] [-r] [n] [keep m]
[advanced_args]
```

if

Conditional execution control structure.

```
de_shell:
if ( expression ) {
then-statement-block
}

de_shell-t:
if ( expression ) {
then-statement-block
}
```

ignore_array_loop_precedences

Removes loop-carried dependencies between array accesses for the Behavioral Compiler product.

```
int ignore_array_loop_precedences
[-process process_name] operations
```
ignore_array_precedences
Removes dependency-related constraints between array accesses for the Behavioral Compiler product.

```
int ignore_array_precedences
[-process process_name]
-from_set from_operations
-to_set to_operations
```

ignore_memory_loop_precedences
Removes loop-carried dependencies between memory accesses for the Behavioral Compiler product.

```
int ignore_memory_loop_precedences
[-process process_name] operations
```

ignore_memory_precedences
Removes dependency related constraints between memory accesses for the Behavioral Compiler product.

```
int ignore_memory_precedences
[-process process_name]
-from_set from_operations
-to_set to_operations
```

include
Executes a script of dc_shell commands. For use in standard dc_shell (desh mode) only.

```
int include file_name [-quiet]
```

incr (dctl-mode only)
Built-in Tcl command.

index_collection
Given a collection and an index into it, if the index is in range, extracts the object at that index and creates a new collection containing only that object. The base collection remains unchanged. For use in dc_shell-t (Tcl mode of dc_shell) only.

```
string index_collection collection index
```
info (dctcl-mode only)
Built-in Tcl command.

insert_bsd
Creates ANSI/IEEE Std 1149.1-compliant boundary scan circuitry using DesignWare macro cells.
int insert_bsd

insert_dft
Adds test points and scan chains to the current design.

insert_jtag
Adds boundary-scan test circuitry to a design.

insert_pads
Inserts I/O pads in a design.
int insert_pads [-verify] [-verify_hierarchically] [-verify_effort low | medium | high] [-thru_hierarchy] [-respect_hierarchy] [design_list]

insert_scan
Adds scan circuitry to a design.
**interp** (dctcl-mode only)
Built-in Tcl command.

**is_false**
Tests the value of a specified variable, and returns a 1 if the value is 0 or the case-insensitive string is false; returns a 0 if the value is 1 or the case-insensitive string is true. For use in Tcl-based dc_shell only.

```
int is_false value
```

**is_true**
Tests the value of a specified variable, and returns a 1 if the value is 1 or the case-insensitive string is true; returns a 0 if the value is 0 or the case-insensitive string is false. For use in Tcl-based dc_shell only.

```
int is_true value
```

**join** (dctcl-mode only)
Built-in Tcl command.

**lappend** (dctcl-mode only)
Built-in Tcl command.

**lib2saif**
Creates a SAIF forward-annotation file from a specified library file or from a list of library files.

```
int lib2saif [-output file_name]
[-lib_pathname lib_path_name]
library_file_name
```

**license_users**
Lists the current users of the Synopsys licensed features.

```
license_users [feature_list]
```

**lindex** (dctcl-mode only)
Built-in Tcl command.
**link**

Resolves design references.

```bash
int link
```

**linsert** (dctcl-mode only)

Built-in Tcl command.

**list**

In standard dc_shell (dcsh mode), lists information about the commands, variables, and licenses currently in Design Analyzer or standard dc_shell (dcsh mode). In Tcl-based dc_shell, it creates a list.

**dc_shell:**

```bash
int list {-commands | -variables
variable_group |
variable_name | -licenses | -files}
```

**dc_shell-t:**

```bash
list ? arg1 arg2 arg arg ... ?
```

**list_attributes**

Lists currently defined attributes. For use in dc_shell-t (Tcl mode of dc_shell) only.

```bash
string list_attributes [-application]
[-class class_name]
```

**list_designs**

List the designs available in dc_shell.

```bash
int list_designs [design_list] [-show_file]
```

**list_files**

Lists the files that are loaded into dc_shell. For use in Tcl-based dc_shell only.

```bash
int list_files
```
list_instances
Lists the instances in dc_shell.

```c
int list_instances [instance_list] [-hierarchy] [-max_levels num_levels] [-full]
```

list_libs
Lists the available libraries in dc_shell.

```c
int list_libs [lib_list]
```

list_licenses
Displays a list of licenses currently checked out by the user. For use in dc_shell -tcl mode only.

```c
string list_licenses
```

llength (dctcl-mode only)
Built-in Tcl command.

lminus (dctcl-mode only)
Built-in Tcl command.

load_of
Returns the capacitance of the specified library cell pin.

```c
float load_of library_cell_pin
```

lrange (dctcl-mode only)
Built-in Tcl command.

lreplace (dctcl-mode only)
Built-in Tcl command.

ls
Lists the contents of a directory.

```c
int ls [filename]
```

lsearch (dctcl-mode only)
Built-in Tcl command.
lsort (dctcl-mode only)
    Built-in Tcl command.

man (dctcl-mode only)
    Built-in Tcl command.

merge_saif
    Reads a list of SAIF files with their corresponding
    weights, annotates switching activity attributes with
    merged toggle_rate and merged static_probability for
    nets, pins, and ports in the current design, and
    generates a merged output SAIF file.

    merge_saif -input_list
    saif_file_and_weight_list
    -instance_name inst_name
    [-output merged_saif_name]
    [-simple_merge]
    [-ignore ignore_name]
    [-ignore_absolute ig_absolute_name]
    [-exclude exclude_file_name]
    [-exclude_absolute ex_absolute_file_name]
    [-unit_base unit_value] [-scale scale_value]
    [-khrate khrate_value]

minimize_fsm
    Performs state minimization on a state table design.

    int minimize_fsm

model
    Creates a model of a design; optionally adds the
    model to a library.

    int model library_name [-design design_name]
    [-output file_name] [-overwrite]
    [-permanent]

namespace (dctcl-mode only)
    Built-in Tcl command.

open (dctcl-mode only)
    Built-in Tcl command.
**optimize_bsd**
Optimizes ANSI/IEEE Std 1149.1-compliant boundary-scan circuitry synthesized by the insert_bsd command using DesignWare macro cells.

`int optimize_bsd`

**optimize_registers**
Performs Behavioral Retiming (BRT) on a mapped gate-level netlist; determines the placement of registers in a design to achieve a target clock period, and minimizes the registers while maintaining that clock period.


**package** (dctcl-mode only)
Built-in Tcl command.

**parent_cluster**
Returns the name of the parent cluster of the passed cluster.

`string parent_cluster cluster_object`

**parse_proc_arguments** (dctcl-mode only)
Built-in Tcl command.
partition_dp
Transforms arithmetic operators (for example, addition, subtraction, and multiplication) into datapath blocks to be implemented by a datapath generator for the specified design or for the current design.

int partition_dp [design_name]
[-duplicate] [-dont_split]

pid (dctcl-mode only)
Built-in Tcl command.

pipeline_design
Pipelines combinational designs by adding registers at the outputs and retiming the circuit.

int pipeline_design design_name
[-period period_value]
[-flatten] [-stages number_of_stages ]
[-stall_ports port_list]
[-stall_polarity high | low]
[-sync_reset reset_port | -async_reset reset_port]
[-reset_polarity high | low]
[-clock_port_name clock_port]
[-no_incremental_map] [-check_design [-verbose]] [-print_critical_loop]
[-no_clock_correction]
[-minimum_period_only] [-register_outputs]

pipeline_loop
Specifies the name of a loop for Behavioral Compiler to pipeline.

int pipeline_loop loop_name
-initiation_interval interval_time
-latency latency_time

plot
Plots the schematic or symbol view for the current design using PostScript format.

int plot [-hierarchy]
[-sheet_list sheet_number_list]
[-output file_name] [-symbol_view]
[-schematic_view]
preschedule
Directs Behavioral Compiler to schedule an operation or operations in a specific cycle for a specified process or for all processes.

int preschedule [-process process_name] {operation_names} cycle

preview_bsd
Previews the boundary-scan design.

int preview_bsd
[-show cells | data_registers | instructions | tap | all]
[-script]

preview_dft
Previews, but does not implement, the test points and scan chains that will be added to the current design, based on the current set of scan and test point specifications.

int preview_dft
[-script]
[-show bidirectionals | cells | scan | scan_clocks | scan_signals | segments | tristates | all]
[-test_points all] [-no_scan]

preview_scan
Previews the scan design.

int preview_scan
[-command insert_scan | reoptimize_design]
[-script]
[-show bidirectionals | cells | scan | scan_clocks | scan_signals | segments | tristates | all]

print_suppressed_messages (dctcl-mode only)
Built-in Tcl command.

48 Synthesis Commands
print_variable_group
Lists the variables defined in a specified variable group, along with their current values. For use in Tcl-based dc_shell only.

int print_variable_group variable_group

printenv (dctcl-mode only)
Built-in Tcl command.

printvar (dctcl-mode only)
Built-in Tcl command.

proc (dctcl-mode only)
Built-in Tcl command.

proc_args (dctcl-mode only)
Built-in Tcl command.

proc_body (dctcl-mode only)
Built-in Tcl command.

propagate_constraints
Propagates timing constraints from lower levels of the design hierarchy to the current design.

int propagate_constraints
[-design design_list]
[-clocks]
[-gate_clock]
[-false_path]
[-multicycle_path]
[-max_delay]
[-min_delay]
[-disable_timing]
[-all]
[-ignore_through_port_exceptions]
[-ignore_from_or_to_port_exceptions]
[-verbose]
[-dont_apply]
[-output output_file_name]

puts (dctcl-mode only)
Built-in Tcl command.
**pwd**
Displays the pathname of the present working
directory (pwd), also called the current directory.

```
int pwd
```

**query_objects**
Searches for and displays objects in the database. For
use in dc_shell-t (Tcl mode of dc_shell) only.

```
string query_objects [-verbose]
[-class class_name]
[-truncate elem_count]
object_spec
```

**quit**
Exits dc_shell

```
int quit
```

**read**
In standard dc_shell (dcsh mode), reads designs into
memory in Synopsys internal database (.db) format.
In Tcl-based dc_shell (dctcl mode), reads from a
channel. In lc_shell, reads libraries into memory in
Synopsys internal database (.db) format.

**dcsh:**
```
list read [-define macro_names]
[-format input_format]
[-single_file file_name]
[-names_file name_mapping_files]
[-define define_list]
file_list
```

**dctcl:**
```
read ?-nonewline? channelId
read channelId numBytes
```

**lc_shell:**
```
list read file_list
```
read_bsd_init_protocol
Reads a boundary-scan initialization protocol file.

int read_bsd_init_protocol [init_protocol_file]

read_bsd_protocol
Reads a custom boundary-scan protocol file.

int read_bsd_protocol [protocol_file]

read_clusters
Annotates a design with physical cluster hierarchy data from a file in Physical Design Exchange Format (PDEF).

int read_clusters [-design design_name] [cluster_file_name]

read_db
Reads in one or more design or library files in Synopsys database (.db) format.

string read_db [file_names]

read_edif
Reads in one or more design or library files in EDIF format.

string read_edif [file_names]

read_file
Reads designs or libraries into dc_shell, or reads libraries into lc_shell.

dc_shell:
list read_file [-define macro_names] [-format input_format] [-single_file file_name] [-names_file name_mapping_files] file_list

lc_shell:
list read_file [file_list]
read_init_protocol
Reads an initialization protocol file.

int read_init_protocol [init_protocol_file]

read_lib
Reads a technology library, physical library, or symbol library into dc_shell or lc_shell.

int read_lib [-format format_name]
[-symbol intermediate_symbol_library_file]
[-file_name file_name] [-no_warnings]
[-names_file file_list]

read_partition
Reads the database for a design.

read_partition
-source pass
-type pre | post
-design_name

read_pin_map
Reads in a port-to-pin mapping file, which defines the design port-to-package pin mapping for a boundary-scan design.

int read_pin_map path_name

read_preserved_function_netlist
Reads an external precompiled netlist for a preserved function.

int read_preserved_function_netlist
[preserved_functions]
[-exclude preserved_functions]
[-design_library library_name]
[-force_reload]
[-filename filename]
[-return_port return_port_name]
[-clock_port_name clock_port_name]
[-clock_edge positive | negative]
[-sync_reset reset_port_name]
[-async_reset reset_port_name]
[-reset_polarity high | low]
read_saif
Reads a SAIF file and annotates the switching activity attributes toggle_rate and static_probability for nets, pins, and ports in the current design.

int read_saif -input file_name
-instance_name name
[-ignore ignore_name]
[-ignore_absolute ig_absolute_name]
[-exclude exclude_file_name]
[-exclude_absolute ex_absolute_file_name]
[-names_file name_changes_log_file]
[-unit_base unit_value] [-scale scale_value]
[-khrate khrate_value]
[-rtl_direct] [-verbose]

read_sdc
Reads in a script in Synopsys Design Constraints (SDC) format.

int read_sdc file_name [-echo]
[-version sdc_version]

read_sdf
Reads leaf cell and net timing information from a file in Standard Design Format (SDF) and uses that information to annotate the current design.

string read_sdf [-load_delay net | cell]
[-path path_name]
[-min_type sdf_min | sdf_typ | sdf_max]
[-max_type sdf_min | sdf_typ | sdf_max]
[-worst]

sdf_file_name

read_test_protocol
Reads a custom test protocol file.

int read_test_protocol [protocol_file]
**read_toggle**
Reads a TOGGLE file and annotates the switching activity attributes toggle_rate and static_probability onto nets, pins, and ports in the database file of the current design.

```
int read_toggle  -input file_name
                 -instance_name name
                 [-unit_base unit_value]
                 [-scale scale_value] [-khrate khrate_value]
                 [-names_file change_name_log_file]
                 [-case_insensitive]
                 [-output saif_file_name]
```

**read_verilog**
Reads in one or more design or library files in Verilog format.

```
string read_verilog file_names
```

**read_vhdl**
Reads in one or more design or library files in VHDL format.

```
string read_vhdl file_names
```

**redirect** *(dctcl-mode only)*
Built-in Tcl command.

**reduce_fsm**
Reduces the logic for each state transition in a state table design.

```
int reduce_fsm
```

**regexp** *(dctcl-mode only)*
Built-in Tcl command.

**regsub** *(dctcl-mode only)*
Built-in Tcl command.
**remove_analysis_info**
Removes BCView analysis information from the specified design.

```plaintext
int remove_analysis_info [design]
```

**remove_annotated_check**
Removes annotated timing check information.

```plaintext
int remove_annotated_check
-all | -from from_list | -to to_list
[-rise | -fall] [-clock rise | fall]
[-setup] [-hold] [-recovery] [-removal]
[-nochange_low] [-nochange_high]
```

**remove_annotated_delay**
Removes the annotated delay between two pins.

```plaintext
int remove_annotated_delay
-all | -from from_list | -to to_list
```

**remove_annotated_transition**
Removes the annotated transition at a pin.

```plaintext
int remove_annotated_transition -all | -at pin_list
```

**remove_attribute**
Removes an attribute from a design or library object.

```plaintext
list remove_attribute object_list
attribute_name [-bus] [-quiet]
```

**remove_bsd_instruction**
Removes boundary-scan instructions from the instruction list to be used by insert_bsd for the current design.

```plaintext
int remove_bsd_instruction instruction_list
```

**remove_bsd_port**
Removes from specified ports the attributes that identify those ports as IEEE Std 1149.1 test access ports (TAPs) in the current design.

```plaintext
int remove_bsd_port port_list
```
remove_bsd_signal
Removes boundary-scan signal types from specified ports in the current design.

`int remove_bsd_signal port_list`

remove_bsdSpecification
Removes the IEEE 1149.1 specifications from a boundary-scan design.


remove_bsr_cell_type
Removes the boundary-scan cell type specification from a set of design ports.

`int remove_bsr_cell_type port_list`

remove_bus
Removes a port bus or net bus.

`int remove_bus object_list`

remove_cache
Selectively removes elements from the synthetic library cache directories.


remove_cell
Removes cells from the current design.

`int remove_cell cell_list | -all`
remove_cell_degradation
Removes the cell_degradation attribute on specified ports or designs.
int remove_cell_degradation object_list

remove_clock
Removes clocks from the current design.
int remove_clock clock_list | -all

remove_clock_gating_check
Removes setup and hold checks from the specified clock gating cells.
int remove_clock_gating_check [-setup] [-hold] object_list

remove_clock_latency
Removes clock latency information from the specified objects.
string remove_clock_latency [-source] object_list

remove_clock_transition
Removes clock transition attributes on the specified clock objects.
int remove_clock_transition clock_list

remove_clock_uncertainty
Removes clock uncertainty information.
string remove_clock_uncertainty object_list
[-from from_clock] [-to to_clock] [-rise] [-fall] [-setup] [-hold]

remove_clusters
Removes the physical cluster hierarchy associated with a design.
int remove_clusters [designs_or_clusters]
remove_constraint
  Removes all constraint attributes, clocks, and path delay information from the current design.
  
  int remove_constraint -all

remove_design
  Removes a list of designs or libraries from dc_shell memory.
  
  int remove_design
    [[design_list] [-hierarchy] | -designs | -all] [-quiet]

remove_dft_configuration
  Removes the current DFT configuration from the current design.
  
  int remove_dft_configuration

remove_dft_signal (dctcl-mode only)
  Built-in Tcl command.

remove_driving_cell
  Removes driving cell attributes from the specified input or inout ports of the current design.
  
  int remove_driving_cell [port_list]

remove_from_collection
  Removes objects from a collection, resulting in a new collection. The base collection remains unchanged. For use in dc_shell-t (Tcl mode of dc_shell) only.
  
  string remove_from_collection collection object_spec

remove_generated_clock
  Removes a generated_clock object.
  
  string remove_generated_clock -all | clock_list
remove_highlighting
- Removes highlighting from schematics of the current instance or the current design.
  - int remove_highlighting [-all | -hier]

remove_ideal_net
- Removes the ideal_net attribute from the specified nets in the current design.
  - int remove_ideal_net net_list

remove_input_delay
- Removes input delay on pins or input ports.

remove_license
- Removes a licensed feature.
  - int remove_license feature_list

remove_multibit
- Removes multibit components in the current design, or detaches cells in the current design from the multibit components that contain them.
  - int remove_multibit object_list

remove_net
- Removes nets from the current design.
  - int remove_net net_list | -all

remove_operand_isolation
- Removes the isolation logic that Power Compiler inserted in the circuit during operand isolation.
  - int remove_operand_isolation [-from from_list] [-to to_list]
remove_output_delay
Removes output delay on pins or output ports.

int remove_output_delay
[-clock clock_name [-clock_fall]
[-level_sensitive]]
[-rise] [-fall] [-max] [-min] port_pin_list

remove_pads
Removes I/O pads from a design.

int remove_pads [-verify]
[-verify_effort effort] [design_list]

remove_pass_directories
Removes the data directories associated with the specified passes.

remove_pass_directories pass_list

remove_pin_map
Removes a design port-to-package pin mapping for a boundary-scan design.

int remove_pin_map package_name

remove_port
Removes ports from the current design.

int remove_port port_list | -all

remove_port_configuration
Removes the Shadow LogicDFT port configuration from the specified port on the specified list of elements, so that insert_dft inserts the default test point circuitry.

int remove_port_configuration
-cell cell_design_ref_list
-port port_name

remove_propagated_clock
Removes a propagated clock specification.

string remove_propagated_clock object_list
remove_rtl_load
   Removes an rtl load value for capacitance and resistance from pins, ports, and nets.
   int remove_rtl_load [-all] [pin_net_list]

remove_scan_register_type
   Removes existing scan register types, previously set by set_scan_register_type, from specified cells or from the current design.
   int remove_scan_register_type
      [cell_or_design_list]

remove_scan_specification
   Removes scan specifications from the current design.
   int remove_scan_specification
      [-all] [-bidirectionals]
      [-chain chain_name_list]
      [-configuration] [-link link_name_list]
      [-segment segment_name_list]
      [-signal port_name_list] [-tristates]

remove_scheduling_constraints
   Removes explicit Behavioral Compiler scheduling constraints from the specified process or from all processes.
   int remove_scheduling_constraints
      [-process process_name]

remove_unconnected_ports
   Removes unconnected ports or pins from cells, references, and subdesigns.
   int remove_unconnected_ports cell_list
      [-blast_buses]

remove_variable
   Removes a variable from dc_shell.
   int remove_variable variable_name
**remove_wire_load_min_block_size**
Removes the wire_load_min_block_size attribute from the current design.

```plaintext
int remove_wire_load_min_block_size
```

**remove_wire_load_model**
Removes wire load model attributes from designs, ports, and hierarchical cells of the current design, or the specified cluster of the current design.

```plaintext
int remove_wire_load_model
[-min] [-max] [-cluster cluster_name]
[object_list]
```

**remove_wire_load_selection_group**
Removes wire load model selection group from designs and cells, or from a specified cluster of the current design.

```plaintext
int remove_wire_load_selection_group [-min]
[-max]
[-cluster cluster_name] [object_list]
```

**remove_wrapper_element**
Removes the wrapper_element attribute from a list of elements.

```plaintext
int remove_wrapper_element
cell_design_ref_list
```

**rename_design**
Renames a design in dc_shell, or moves a list of designs to a file.

```plaintext
int rename_design design_name1 design_name2
```
**reoptimize_design**  
Incrementally optimizes a design using layout/floorplanning information.

```
int reoptimize_design
[-map_effort low | medium | high]
[-sizing] [-pin_swap] [-buffer_insertion]
[-buffer_removal]
[-cpr]
[-tolerance_to_change low | medium | high]
[-verify]
[-verify_hierarchically]
[-verify_effort low | medium | high]
[-ignore_footprint] [-ignore_cell_area]
[-no_design_rule | -only_design_rule]
[-area_recovery]
[-reorder_scan_chains] [-no_reoptimize]
```

**replace_fpga**  
Replaces field-programmable cells in the current design with gates in the target library.

```
int replace_fpga [-force] [-group_cells]
[-group_tlus]
[verify [-verify_hierarchically]
[-verify_effort low | medium | high]]
```

**replace_synthetic**  
Implements all synthetic library parts of a design using generic logic.

```
int replace_synthetic [-ungroup]
```

**report_annotated_check**  
Displays all annotated timing checks on the current design.

```
int report_annotated_check [-nosplit]
```

**report_annotated_delay**  
Displays all annotated delays on cells and nets of the current design.

```
int report_annotated_delay [-cell] [-net]
[-min] [-nosplit]
```
report_annotated_transition
Displays annotated transitions on all pins of the current design.

int report_annotated_transition

report_area
Displays area information and statistics for the design of the current instance, if set; or for the current design otherwise.

int report_area [-nosplit] [-physical]

report_attribute
Displays attributes and their values associated with a cell, net, port, instance, or design.


report_budget
Displays budgeting information about a design. The delay number shown for each cell shows the amount of budgets allocated for that cell in the path. The overall budget for the path is the sum of these cell budgets (delays).

report_buffer_tree
Displays the buffer tree and its level information at the given driver pin.

int report_buffer_tree [-from start_point_list | -net net_list] [-depth max_depth] [-connections] [-nosplit]

report_bus
Lists the bused ports and nets in the current instance, if set; or in the current design otherwise.

int report_bus [-nosplit]

report_cache
Reports on the contents of synthetic library caches.


report_cell
Displays information about cells in the current instance, if set; or in the current design otherwise.

int report_cell [-nosplit] [-connections [-verbose]] [-physical [-verbose]] [cell_list]

report_clock
Displays clock-related information on the current design.

int report_clock [-attributes] [-skew] [-nosplit]
report_clock_gating
Reports information for clock gating cells, and gated
and ungated registers of the current design.

report_clock_gating [-gating_elements]
[-gated]
[-ungated] [-only <cell_name_list>]
[-hier] [-verbose] [-no_split]

report_clusters
Reports on the physical cluster hierarchy associated
with the current design.

int report_clusters [-cluster cluster_name]
[-leaf] [-nosplit]

report_compile_options
Displays information about the compile options for
the design of the current instance, if set; or for the
current design otherwise.

int report_compile_options [-nosplit]

report_constraint
Displays constraint-related information about a
design.

int report_constraint [-all_violators]
[-verbose]
[-significant_digits digits]
[-max_area] [-max_delay] [-critical_range]
[-min_delay]
[-max_capacitance] [-min_capacitance]
[-max_transition]
[-max_fanout] [-cell_degradation]
[-min_porosity]
[-max_dynamic_power] [-max_leakage_power]
[-connection_class] [-multiport_net]
[-nosplit]
**report_delay_calculation**
Displays the actual calculation of a timing arc delay value for a cell or net.

```shell
int report_delay_calculation
  -min
  -max
  -from from_pin
  -to to_pin
  [-nosplit]
```

**report_design**
Displays attributes on the current design.

```shell
int report_design [-nosplit] [-physical]
```

**report_design_lib**
Lists the design units contained in the specified libraries.

```shell
int report_design_lib [-libraries]
  [-designs] [-architectures] [-packages]
  [library_list]
```

**report_fpga**
Reports information about resource usage for supported FPGA architecture.

```shell
int report_fpga [-one_level] [-nosplit]
```

**report_fsm**
Displays state-machine attributes and information for the design of the current instance, if set; or for the current design otherwise.

```shell
int report_fsm [-nosplit]
```

**report_hierarchy**
Displays the reference hierarchy of the current instance, if set; or of the current design otherwise.

```shell
int report_hierarchy [-nosplit] [-full]
```
**report_internal_loads**
Displays internal loads on the nets in the current design.

```plaintext
int report_internal_loads [-nosplit]
```

**report_lib**
Displays information about technology or symbol libraries.

```plaintext
int report_lib library_name [-timing_arcs]
[-timing]
[-power] [-em] [-vhdl_name] [-table]
[-full_table]
[-timing_label] [-power_label] [-all]
[cell_list]
```

**report_multibit**
Displays information about multibit components in the current design.

```plaintext
int report_multibit [-nosplit] [object_list]
```

**report_multicycles**
Reports on scheduled multicycle operations for Behavioral Compiler.

```plaintext
int report_multicycles
[-process process_name]
```

**report_name_rules**
Reports the values of name rules.

```plaintext
int report_name_rules [name_rules]
```

**report_names**
Reports potential name changes of ports, cells, and nets in a design.

```plaintext
int report_names [-rules name_rules]
[-hierarchy] [-original] [-nosplit]
```
**report_net**
Displays net information for the design of the current instance, if set; or for the current design otherwise.

```bash
```

**report_operand_isolation**
Reports the status of operand isolation cells in the current design

```bash
int report_operand_isolation
```

**report_packages**
Displays the package names for all the port-to-pin mapping files that are read in for a boundary-scan design.

```bash
int report_packages
```

**report_pass_data**
Reports the data files that are available for a design created by Automated Chip Synthesis.

```bash
int report_pass_data [-hierarchy] [design] [-pass_list pass_list]
```

**report_path_group**
Reports information about path groups in the current design.

```bash
int report_path_group [-nosplit]
```

**report_port**
Displays information about ports for the design of the current instance, if set; or for the current design otherwise.

```bash
int report_port [-drive] [-verbose] [-physical] [-nosplit] [port_list]
```
**report_power**
Calculates and reports dynamic and static power for a design or instance.

```lua
int report_power [-net] [-cell]
[-only cell_or_net_list]
[-hier] [-hier_level level_value]
[-verbose] [-cumulative] [-flat]
[-exclude_boundary_nets]
[-analysis_effort low | medium | high]
[-nworst number] [-sort_mode mode]
[-histogram
 [-exclude_leq le_val | -exclude_geq ge_val]]
[-nosplit]
```

**report_reference**
Displays information about references in the current instance, if set; or in the current design otherwise.

```lua
int report_reference [-nosplit]
```

**report_resource_estimates**
Displays timing and area estimates for Behavioral Compiler operations in the current design.

```lua
int report_resource_estimates
```

**report_resources**
Lists the resources used in the design of the current instance, if set; or in the current design otherwise.

```lua
int report_resources [-nosplit]
```

**report_routability**
Displays information about the routability of the current design.

```lua
int report_routability [-nosplit]
```

**report_saif**
Reports switching activity annotations for nets, pins, and ports of the current design or instance.

```lua
int report_saif [-flat] [-type rtl | gate]
[-missing] [-only cell_or_net_list]
```
report_schedule
Displays the results of scheduling and allocation as performed by Behavioral Compiler.

```c
int report_schedule [-process process_name]
[-operations [-mask [r][w][l][L][o][p]]
[-start start_cycle]
[-finish end_cycle] [-delimiter character]]
[-variables [-min min_width]
[-max max_width]
[-abstract_fsm [-mask [r][w][o][s]]]
[-summary]
[-abstract_fsm [-mask [r][w][o][d][s]]]
```

report_scheduling_constraints
Displays Behavioral Compiler scheduling constraints on the current design for a specified process or for all processes.

```c
int report_scheduling_constraints
[-process process_name] [-dont_chain]
[-chains] [-antichains] [-two_point]
[-preschedule] [-pipeline]
```

report_synlib
Displays information about synthetic libraries.

```c
int report_synlib library [module_list]
```
**report_test**

Displays test-related information about the current design.

```
int report_test [-assertions] [-atpg_conflicts] [-bsd]
[-bsd_configuration] [-configuration] [-constraints] [-coverage] [-dft]
[-incremental] [-inst design_instance_list] [-dont_fault]
[-faults [-class abandoned | detected | redundant]
  | tied | untested | hyperactive | oscillating | probable | no_status | dont_fault] [-incremental]
[-inst design_instance_list] [-jtag]
[-mask_fault [-inst design_instance_list]]
[-methodology] [-port] [-scan_path] [-state]
[-testsim_timing]
[-trace_nets] [-nosplit]
```

**report_timing**

Displays timing information about a design.

```
int report_timing
[-to to_list] [-from from_list]
[-through through_list]
[-delay min | min_rise | min_fall | max | max_rise | max_fall]
[-nworst paths_per_endpoint]
[-max_paths max_path_count]
[-input_pins] [-nets] [-transition_time]
[-capacitance]
[-locations] [-physical]
[-lesser_path max_path_delay]
[-greater_path min_path_delay] [-loops]
[-true [-true_threshold path_delay]]
[-justify] [-enable_preset_clear_arcs]
[-significant_digits digits] [-nosplit]
[-sort_by group | slack]
```
**report_timing_requirements**
Reports timing path requirements (user attributes) and related information.

```bash
int report_timing_requirements
  [-attributes][-ignored]
  [-from from_list][-through through_list]
  [-to to_list][-expanded][-nosplit]
```

**report_transitive_fanin**
Reports logic in the transitive fanin of specified sinks.

```bash
int report_transitive_fanin -to sink_list [-nosplit]
```

**report_transitive_fanout**
Reports logic in the transitive fanout of specified sources.

```bash
int report_transitive_fanout
  -clock_tree | -from source_list [-nosplit]
```

**report_ultra_optimization**
Reports on whether or not the DC Ultra optimization mode is set and whether licenses have been checked out correctly.

```bash
int report_ultra_optimization
```

**report_wire_load**
Displays the characteristics of the wire load models set on a design or in a library.

```bash
int report_wire_load
  [-design design_name]
  [-name model_name] [-libraries] [-nosplit]
```

**report_xref**
Generates a cross-reference between schematic objects and sheets on which they occur.

```bash
int report_xref [-nosplit]
```

**reset_compare_design_script**
Removes the compare_design script if it exists.

```bash
int reset_compare_design_script
```
reset_design
   Removes from the current design all user-specified objects and attributes, except those defined using set_attribute.

   int reset_design

reset_path
   Resets specified paths to single cycle timing.

   int reset_path [-setup | -hold]
   [-rise | -fall]
   [-from from_list] [-through through_list]
   [-to to_list]

reset_switching_activity
   Removes the toggle_rate and static_probability attributes, and/or the max_toggle_rate attribute, from nets, pins, cells and/or ports of the current design.

   int reset_switching_activity
   [-switching_activity] | [-max_toggle_rate] | [-all]
   [-verbose]

restore_test
   Restores the specified test program to dc_shell.

   string restore_test test_program_name

return (dctcl-mode only)
   Built-in Tcl command.

rtl2saif
   Creates a SAIF forward-annotation file starting from the top level of the design.

   int rtl2saif [-output file_name]
   [-design design_name]
rtl_analyzer

Invokes RTL Analyzer on the current design.

```
int rtl_analyzer [-output out_db_file]
[-project_file project_file_name]
[-search_additional path_list]
[-dont_start]
[-host machine_name]
[-arch architecture_name]
[-use_primetime]
```

rtlrc

Analyzes the testability of a design at the RTL and gate level, using Verilog or VHDL RTL sources.

```
int rtlrc [-max_detail_lines n]
```

scan (dctcl-mode only)

Built-in Tcl command.

schedule

Invokes the scheduling and allocation functions of Behavioral Compiler.

```
int schedule
[-effort zero | low | medium | high]
[-io_mode cycle_fixed | superstate_fixed]
[-extend_latency]
[-host hostname]
[-arch remote_host_architecture]
[-allocation_effort zero | low | medium | high]
```

seek (dctcl-mode only)

Built-in Tcl command.

set (dctcl-mode only)

Built-in Tcl command.
set_annotated_check
Sets the setup, hold, recovery, or removal timing check value between two pins.

```
int set_annotated_check check_value
   -from from_pins -to to_pins
   -setup | -hold | -recovery | -removal |
   -nochange_high |
   -nochange_low [-rise | -fall]
   [-clock clock_check] [-worst]
```

set_annotated_delay
Sets the net or cell delay value between two pins.

```
int set_annotated_delay -net | -cell
   [-load_delay load_delay_type]
   [-rise | -fall] [-min] [-max] delay_value
   -from from_pins -to to_pins [-worst]
```

set_annotated_transition
Sets the transition time at a given pin.

```
int set_annotated_transition
   [-rise | -fall] [-min] [-max] transition
port_pin_list
```

set_attribute
Sets the value of an attribute on a design or library object.

```
list set_attribute object_list
   attribute_name
   attribute_value
   [-type boolean | integer | float | string]
   [-bus] [-quiet]
```

set_auto_ideal_nets
Sets the auto_ideal_net attribute on the current design, causing the specified networks to be treated as ideal nets.

```
int set_auto_ideal_nets [-default] [-none]
   [-all] [-clock true | false]
   [-constant true | false]
```
set_autofix_clock
If Autofix is enabled, specifies a clock port to be used for specified cells, for automatic fixing of uncontrollable clock violations during execution of preview_dft or insert_dft.

```
int set_autofix_clock clock_port cell_list
```

set_autofix_configuration
If Autofix is enabled, optionally disables automatic fixing of all uncontrollable clock violations or all asynchronous preset/clear violations, during execution of preview_dft or insert_dft.

```
int set_autofix_configuration [-clock true | false -async true | false]
```

set_autofix_element
If Autofix is enabled, optionally disables automatic fixing of uncontrollable clock violations or asynchronous preset/clear violations for specified sequential elements, during execution of preview_dft or insert_dft.

```
int set_autofix_element cell_listf -clock true | false -async true | false
```

set_balance_registers
Sets the balance_registers attribute on the specified designs or on the current design, so that the design is retimed during compile.

```
int set_balance_registers [true | false]
[-design design_list]
```

set_behavioral_reset
Directs Behavioral Compiler to set reset behavior for process, port, synchronicity, active state, direct connection or FSM generation.

```
```
set_boundary_optimization
Sets the boundary_optimization attribute on specified cells, references, or designs, thus allowing for optimization across hierarchical boundaries.

```
int set_boundary_optimization obj_list
[true | false]
```

set_bsd_bsr_element
Characterizes a design cell as a boundary-scan register to be used for the boundary-scan insertion.

```
int set_bsd_bsr_element -type cell_type
-design design_name
-access access_list
```

set_bsd_compliance
Specifies an IEEE 1149.1 compliance-enable pattern for a boundary-scan design.

```
int set_bsd_compliance
signal_port_bit_value_pairs
```

set_bsd_configuration
Specifies the boundary-scan configuration for a design.

```
int set_bsd_configuration
[-asynchronous_reset true | false]
[-default_package package_name]
[-instruction_encoding default | one_hot]
[-ir_width instruction_register_length]
[-style asynchronous | synchronous]
```

set_bsd_control_cell
Declares a boundary-scan control register cell.

```
int set_bsd_control_cell
BSD_control_register_name -type cell_type
-port_list port_list
```
set_bsd_data_cell
Specifies a boundary-scan register cell type to be used on a specified list of ports in the current design.

    int set_bsd_data_cell cell_type
- port_list port_list
[-direction in | out]

set_bsd_instruction
Specifies boundary-scan instructions to be used by insert_bsd for the current design.

    int set_bsd_instruction instruction_set
[-code inst_code_list]
[-register user_data_reg]
[-input_clock_condition clock_conditioning]
[-output_condition output_conditioning]
[-internal_scan pin_name]

set_bsd_intest
Specifies the parameters for the INTEST instruction.

    int set_bsd_intest [-time real_numbers]
[-clock_cycles clock_port_integer_pairs]

set_bsd_linkage_port
Identifies the linkage ports in your design.

    int set_bsd_linkage_port -port_list list_of_ports

set_bsd_path
Specifies the order of the cells in the boundary scan register.

    int set_bsd_path identifier_list

set_bsd_port
Identifies existing ANSI/IEEE Std. 1149.1 Test Access ports of the current design, for the check_bsd command.

    int set_bsd_port port_type TAP_port
set_bsd_register
Declares a user-defined data register to be used for the boundary-scan insertion.

```bash
int set_bsd_register register_identifier
  -cell hierarchical_cell_name
  -access_list access_list
```

set_bsd_runbist
Specifies the parameters for the RUNBIST instruction.

```bash
int set_bsd_runbist [-time runtime]
  [-clock_cycles clock_port_integer_pairs]
  [-signature pattern]
```

set_bsd_signal
Specifies a boundary scan signal type to be placed on a specified port in the current design.

```bash
int set_bsd_signal port_type port_name
```

set_bsd_tap_element
Characterizes a design cell as a boundary-scan tap controller to be used for the boundary-scan insertion.

```bash
int set_bsd_tap_element -design design_name
  -access access_list
```

set_bsr_cell_type
Specifies the minimum acceptable boundary-scan cell implementation for a set of ports in the current design.

```bash
int set_bsr_cell_type -port port_list
  [-direction in | out] cell_type
```

set_cell_degradation
Sets the cell_degradation attribute to a specified value on specified ports or designs.

```bash
int set_cell_degradation
  cell_degradation_value object_list
```
set_cell_internal_power
Sets or removes the power_value attribute on or from specified pins; the value represents the power consumption for a single toggle of each pin.

```
int set_cell_internal_power pin_names
[power_value [unit]]
```

set_clock_gating_check
Puts setup and hold checks on clock gating cells.

```
int set_clock_gating_check
[-setup setup_margin]
[-hold hold_margin] object_list
```

set_clock_gating_signals
Independent of the conditions necessary for automatic RTL clock gating, flip-flops inferred from HDL signals or variables specified by this command either are implemented or are not implemented with a gated clock.

```
int set_clock_gating_signals
[-design design_name]
[-include signal_list]
[-exclude signal_list]
```

set_clock_gating_style
Sets the clock gating style that HDL Compiler uses for clock gating.

```
int set_clock_gating_style
[-sequential_cell seq_cell]
[-minimum_bitwidth minsize_value]
[-setup setup_value]
[-hold hold_value]
[-positive_edge_logic gate_list]
[-negative_edge_logic gate_list]
[-control_point none | before | after]
[-control_signal scan_enable | test_mode]
[-observation_point true | false]
[-observation_logic_depth depth_value]
[-max_fanout max_fanout_count]
[-no_sharing]
```
set_clock_latency
Specifies clock network latency.


set_clock_transition
Sets clock transition attributes on clock objects.

int set_clock_transition transition [-rise | -fall] [-min] [-max] clock_list

set_clock_uncertainty
Specifies uncertainty (skew) of clock networks.


set_combinational_type
Sets attributes on cell instances to specify which combinational cells from the target library are to be used by compile.

int set_combinational_type -replacement_gate replacement_gate [cell_list]
**set_common_resource**
Specifies a group of operations to be scheduled on the same resources by Behavioral Compiler.

```
int set_common_resource
[-process process_name]
opreation_names [-min_count min_resources]
[-max_count max_resources] [-force_sharing]
[-exclusive]
```

**set_compare_design_script**
A command to be added to the compare_design script, to be used during verification with the balance_buffer, compare_design, compile, insert_pads, reoptimize_design, replace_fpga, and translate commands.

```
int set_compare_design_script
[-ignore endpoint_list]
[-only endpoint_list]
[-accept sub_design_list]
```

**set_compile_partitions**
Specifies the compile entities (partitions) for the current design.

```
set_compile_partitions
(-level level) | (-designs design_list) | -all
[-force]
[-no_reset]
```

**set_connection_class**
Sets the connection class value on ports.

```
int set_connection_class
connection_class_value object_list
```

**set_cost_priority**
Sets the cost_priority attribute to a specified value on the current design.

```
int set_cost_priority [-default] [-delay]
cost_list
```
set_critical_range
Sets the critical_range attribute to a specified value on a list of designs.

int set_critical_range range_value designs

set_cycles
Sets the number of cycles between two Behavioral Compiler operations and/or loop boundaries for a specified process or for all processes.

int set_cycles
[-process process_name] cycle_offset
-from_option start_operation -to_option end_operation

set_default_drive (dctcl-mode only)
Built-in Tcl command.

set_default_driving_cell (dctcl-mode only)
Built-in Tcl command.

set_default_fanout_load (dctcl-mode only)
Built-in Tcl command.

set_default_input_delay
Sets the input delay as a percentage of the clock period to be assigned during environment propagation.

set_default_input_delay
[-none]
percent_delay
[cell_or_pin_list]

set_default_load (dctcl-mode only)
Built-in Tcl command.
set_default_output_delay
Sets the output delay as a percentage of the clock period to be assigned during environment propagation.

```
set_default_output_delay
[-none]
percent_delay
[cell_or_pin_list]
```

set_design_license
Adds license information to the current design. The set_design_license command can be used to require a license before a design can be read in.

```
int set_design_license
[-dont_show references] [-quiet] [-limited limited_keys] regular_keys
```

set_dft_configuration
Sets the DFT configuration for the current design.

```
int set_dft_configuration
[-order list_of_clients]
```

set_dft_signal
Specifies a list of ports for insert_dft to use for implementing test point control signals of a specified signal type.

```
int set_dft_signal
dft_signal_type -port port_list
[-hookup pin]
```

set_disable_timing
Disables timing arcs in the current design.

```
int set_disable_timing object_list
[-from from_pin_name -to to_pin_name]
[-restore]
```
set_dont_touch
Sets the dont_touch attribute on cells, nets, references, and designs in the current design, and on library cells, to prevent these objects from being modified or replaced during optimization.

int set_dont_touch object_list
[true | false]

set_dont_touch_network
Sets the dont_touch_network attribute on clocks, pins, or ports in the current design, to prevent cells and nets in the transitive fanout of the set_dont_touch_network objects from being modified or replaced during optimization.

int set_dont_touch_network object_list

set_dont_use
Sets the dont_use attribute on library cells to exclude them from the target library during optimization.

int set_dont_use object_list

set_drive
Sets the rise_drive or fall_drive attributes to specified resistance values on specified input and inout ports.

int set_drive resistance [-rise] [-fall]
[-min] [-max]
port_list

set_driving_cell
Sets attributes on input or inout ports of the current design, specifying that a library cell or pin will drive the ports.

int set_driving_cell
[-lib_cell lib_cell_name] [-library lib]
[-rise] [-fall] [-pin pin_name]
[-from_pin from_pin_name]
[-dont_scale] [-no_design_rule]
[-input_transition_rise rtranfp]
[-input_transition_fall ftranfp]
[-multiply_by factor]
port_list
**set_eco_align**  
Specifies a pair of objects to align.

```
int set_eco_align type
  [object1_name object2_name]
```

**set_eco_obsolete**  
Specifies a reused cell and optionally a new cell name. At runtime of eco_recycle, the reused becomes a fresh cell with the new cell name which is given as an option to the command. Additionally, an obsolete cell is created which gets the original name of the reused cell.

```
int set_eco_obsolete
  [-set] [-remove] [-print]
  cell_name new_name
```

**set_eco_recycle**  
Specifies a pair of cells, one freshly synthesized and one resource cell. The eco_recycle command tries to replace the fresh cell with the resource cell.

```
int set_eco_recycle
  [-set] [-remove] [-print]
  cell_name_1 cell_name_2
```

**set_eco_reuse**  
Specifies the cells or subdesigns to reuse between netlists.

```
int set_eco_reuse cell_name_1 cell_name_2
```

**set_eco_target**  
Specifies a target object and an associated set of tap objects.

```
int set_eco_target target_object -taps
  tap_objects
```

**set_eco_unique**  
Specifies objects that are unique to a design.

```
int set_eco_unique category type object_name
```
set_electromigration_drc
Sets or resets the electromigration DRC constraint for the current design by setting the electromigration_drc attribute.

\[
\text{int } \text{set_electromigration_drc on | off [use_switching_activity]}
\]

set_equal
Defines two input ports as logically equivalent.

\[
\text{int set_equal port1 port2}
\]

set_exclusive_use
Maps a specified HDL variable to a unique Behavioral Compiler register for a specified process or for all processes.

\[
\text{int set_exclusive_use [-process process_name] variable_name [-shared]}
\]

set_false_path
Removes timing constraints from particular paths.

\[
\text{int set_false_path [-rise | -fall] [-setup | -hold] [-from from_list] [-through through_list] [-to to_list] [-reset_path]}
\]

set_fanout_load
Sets the fanout_load attribute to a specified value on specified output ports of the current design.

\[
\text{int set_fanout_load value port_list}
\]

set_fix_hold
Sets a fix_hold attribute on clocks in the current design.

\[
\text{int set_fix_hold clock_list}
\]
**set_fix_multiple_port_nets**
Sets the fix_multiple_port_nets attribute to a specified value on the current design.

```
int set_fix_multiple_port_nets -default | -all | [-feedthroughs] [-outputs]
[-constants] [-buffer_constants]
```

**set_flatten**
Sets or removes the flatten attribute on specified designs or on the current design, to enable or disable the flattening optimization step during compile.

```
int set_flatten [true | false]
[-effort low | medium | high]
[-minimize single_output | multiple_output | none]
[-phase true | false] [-design design_list]
[-quiet]
```

**set_fsm_encoding**
Specifies the bit encodings for states in the current design.

```
int set_fsm_encoding encoding_list
```

**set_fsm_encoding_style**
Defines the encoding style for assigning unencoded states.

```
int set_fsm_encoding_style
one_hot | binary | gray | auto
```

**set_fsm_minimize**
Determines whether or not state minimization is to be performed on the state machine design during compile.

```
int set_fsm_minimize true | false
```

**set_fsm_order**
Sets the ordering of states in a state machine design.

```
int set_fsm_order state_list
```
set_fsm_preserve_state  
Specifies states to be preserved during state minimization.

int set_fsm_preserve_state state_list

set_fsm_state_vector  
Specifies the instance names for flip-flops used to implement the state vector.

int set_fsm_state_vector vector_list

set_ideal_net  
Sets the ideal_net attribute on specified individual nets in the current design.

int set_ideal_net net_list

set_impl_priority  
Sets the formula attribute of the priority parameter and/or the set_id attribute for implementations in synthetic libraries.

int set_impl_priority [-priority formula] [-set_id id] implementation_list

set_implementation  
Specifies the implementation to use for synthetic library cell instances in a design.

int set_implementation implementation_name cell_list [-check_impl]

set_input_delay  
Sets input delay on pins or input ports relative to a clock signal.

int set_input_delay delay_value 
[-clock clock_name [-clock_fall]
[-level_sensitive]]
[-rise] [-fall] [-max] [-min] [-add_delay] port_pin_list
set_input_transition
Sets the max_transition_rise, max_transition_fall, 
min_transition_rise, or min_transition_fall attributes 
to the specified transition values on the specified 
input and inout ports.

int set_input_transition transition [-rise] 
[-fall] [-min] [-max] 
port_list

set_isolation_operations
Specifies a group of operations for operand isolation 
by Behavioral Compiler.

int set_isolation_operations 
[-process process_name] 
[-exclude_operations operation_names] 
[-include_operations operation_names]

set_jtag_implementation
Specifies the implementation of a boundary-scan 
component used during JTAG synthesis.

int set_jtag_implementation component_type 
[implementation [port_list]] | [-default]

set_jtag_instruction
Specifies a boundary-scan instruction that is 
recognized, decoded, and acted upon by the 
synthesized boundary-scan test circuitry.

int set_jtag_instruction instruction_name 
[-code binary_code] 
[-reg_name register_name] | 
[-cancel]

set_jtag_manufacturer_id
Sets the manufacturer identification number of a set 
of designs.

int set_jtag_manufacturer_id manufacturer_id 
[design_list]
**set_jtag_part_number**
Specifies the part number of a list of designs.

```
int set_jtag_part_number part_number
[design_list]
```

**set_jtag_port**
Specifies the ports to be excluded from or included in the JTAG Boundary Scan Register (BSR).

```
int set_jtag_port [true | false port_list] |
[-default]
```

**set_jtag_port_mode**
Specifies the JTAG operational mode of ports in the boundary-scan register (BSR).

```
int set_jtag_port_mode
[control | observe | both port_list] |
[-default]
```

**set_jtag_port_routing_order**
Specifies the order in which JTAG boundary-scan cells associated with I/O ports are connected in the Boundary Scan Register.

```
int set_jtag_port_routing_order
[ordered_port_list] | [-default]
```

**set_jtag_port_type**
Specifies the signal data type of ports in the Boundary Scan Register (BSR).

```
int set_jtag_port_type [clock | enable | data port_list] | [-default]
```

**set_jtag_version_number**
Specifies the version number of a set of designs.

```
int set_jtag_version_number version_num
[design_list]
```

**set_layer**
Defines features of a schematic layer.

```
int set_layer layer_name attribute value
```
set_libcell_dimensions
Sets the width and height of a library cell.

```
int set_libcell_dimensions
-cell cell_name
-width width
-height height
```

set_libpin_location
Sets the location of a pin of a library cell relative to
the origin of the library cell.

```
int set_libpin_location
-cell library_cell_name
-pin pin_name_of_the_library_cell
-coordinate {x_coordinate y_coordinate}
```

set_load
Sets the load attribute to a specified value on specified
ports and nets.

```
int set_load value objects
[-subtract_pin_load]
[-min] [-max] [[-pin_load] [-wire_load]]
```

set_local_link_library
Sets the local_link_library attribute to specified files
and libraries on the current_design.

```
int set_local_link_library
local_link_library
```

set_logic_dc
Specifies one or more input ports in the current design
that are to be driven by dont_care. The set_logic_one
and set_logic_zero commands are used the same way
as this command.

```
int set_logic_dc port_list
```
set_logic_one
Specifies one or more input ports in the current design that are to be driven by logic 1. The set_logic_zero and set_logic_dc commands are used the same way as this command.

int set_logic_one port_list

set_logic_zero
Specifies one or more input ports in the current design that are to be driven by logic 0. The set_logic_one and set_logic_dc commands are used the same way as this command.

int set_logic_zero port_list

set_map_only
Sets the map_only attribute on specified objects so that they can be excluded from logic-level optimization during compile.

int set_map_only object_list flag

set_max_area
Sets the max_area attribute to a specified value on the current design.

int set_max_area [-ignore_tns] area_value

set_max_capacitance
Sets the max_capacitance attribute to a specified value on the specified ports and designs.

int set_max_capacitance capacitance_value object_list

set_max_cycles
Sets the maximum number of cycles between two Behavioral Compiler operations and loop boundaries for a specified process or for all processes.

int set_max_cycles
[-process process_name] cycle_offset
[from_option start_operation]
[to_option end_operation]
set_max_delay
Specifies a maximum delay target for paths in the current design.

```plaintext
int set_max_delay delay_value
[-rise | -fall]
[-from from_list] [-through through_list]
[-to to_list] [-group_path group_name]
[-reset_path]
```

set_max_dynamic_power
Sets the target dynamic power for the current design by setting the max_dynamic_power attribute to a specified value.

```plaintext
int set_max_dynamic_power dynamic_power
[GW | M\text{W} | K\text{W} | W | m\text{W} | u\text{W} | n\text{W} | p\text{W} | f\text{W} | a\text{W}]
```

set_max_fanout
Sets the max_fanout attribute to a specified value on specified input ports and/or designs.

```plaintext
int set_max_fanout fanout_value object_list
```

set_max_leakage_power
Sets the target leakage power for the current design by setting the max_leakage_power attribute to a specified value.

```plaintext
int set_max_leakage_power leakage_power
[GW | M\text{W} | K\text{W} | W | m\text{W} | u\text{W} | n\text{W} | p\text{W} | f\text{W} | a\text{W}]
```

set_max_time_borrow
Sets the max_time_borrow attribute to a specified value on clocks, latch cells, data pins, or clock (enable) pins to constrain the amount of time borrowing possible for level-sensitive latches.

```plaintext
int set_max_time_borrow delay_value object_list
```
**set_max_toggle_rate**
Sets or resets the max_toggle_rate attribute on designs, cells, nets, pins, and ports of the current design.

```plaintext
int set_max_toggle_rate object_list
    [-value max_tr_value]
    [-clock clock_name]
```

**set_max_transition**
Sets the max_transition attribute to a specified value on specified ports or designs.

```plaintext
int set_max_transition transition_value object_list
```

**set_memory_input_delay**
Sets the input delay on a memory to be used by Behavioral Compiler.

```plaintext
int set_memory_input_delay [delay_value]
    [-external ext_delay_value] [-name mem_name]
```

**set_memory_output_delay**
Sets the output delay on a memory to be used by Behavioral Compiler, and enables operation chaining on the outputs of the memory.

```plaintext
int set_memory_output_delay delay_value
    [-external ext_delay_value] [-name mem_name]
```

**set_min_capacitance**
Sets the min_capacitance attribute to a specified value on specified input ports in the current design.

```plaintext
int set_min_capacitance capacitance_value object_list
```
set_min_cycles
Sets the minimum number of cycles between two
Behavioral Compiler operations and loop boundaries
for a specified process or for all processes.

```
int set_min_cycles
[-process process_name] cycle_offset
-from_option start_operation -to_option
end_operation
```

set_min_delay
Specifies a minimum delay target for paths in the
current design.

```
int set_min_delay delay_value
[-rise | -fall] [-from from_list]
[-through through_list]
[-to to_list] [-reset_path]
```

set_min_fault_coverage
Specifies the minimum acceptable fault coverage for
the design.

```
int set_min_fault_coverage
min_fault_coverage
[-timing_critical] [-area_critical]
```

set_min_library
Sets an alternate library to use for minimum delay
analysis.

```
int set_min_library max_library
-min_version min_library | -none
```

set_min_porosity
Sets the minimum_porosity attribute on specified
designs or on the current design.

```
int set_min_porosity porosity_value
[design_list]
```
set_minimize_tree_delay
Sets the minimize_tree_delay attribute on a design or
designs, thus determining whether an arithmetic
expression tree will be restructured to minimize delay
during compile. By default, all expression trees are
candidates for tree height minimization if timing
constraints are specified.

\[
\text{int set_minimize_tree_delay [true | false]} \\
[-\text{design design_list}]
\]

set_model_drive
Sets the model_drive attribute to a specified value on
specified input or inout ports to set their drive values
during synthetic library modeling.

\[
\text{int set_model_drive drive_value port_list}
\]

set_model_load
Sets the model_load attribute to a specified value on
specified ports to set their load values during
synthetic library modeling.

\[
\text{int set_model_load load_value port_list}
\]

set_model_map_effort
Sets the model_map_effort attribute to a specified
value on the current design, to specify the relative
amount of CPU time to use during synthetic library
modeling.

\[
\text{int set_model_map_effort low | medium | high}
\]

set_model_scale
Sets the model_scale attribute to a specified value on
the current design, to use as a scale factor in
calculating timing constraints during synthetic library
modeling.

\[
\text{int set_model_scale scale}
\]
set_multibit_options
Sets the multibit_mode and minimum_multibit_width attributes to specified values on the current design.

```
int set_multibit_options [-default] [-mode multibit_mode] [-minimum_width width]
```

set_multicycle_path
Modifies the single-cycle timing relationship of a constrained path.

```
int set_multicycle_path path_multiplier [-rise | -fall] [-setup | -hold] [-start | -end] [-from from_list] [-to to_list] [-through through_list] [-reset_path]
```

set_operand_isolation_cell
Specifies a list of GTECH cells to be operand isolation candidates.

```
int set_operand_isolation_cell [object_list]
```

set_operand_isolation_slack
Sets the timing threshold below which the automatic isolation roll back operation is not triggered.

```
int set_operand_isolation_slack slack_number
```

set_operand_isolation_style
Sets the operand isolation style that Power Compiler uses for operand isolation.

```
int set_operand_isolation_style [-logic logic_style]
```
set_operating_conditions
Defines the operating conditions for the current design.

int set_operating_conditions
[-min min_condition] [-max max_condition]
[-min_library min_lib]
[-max_library max_lib]
[-library lib]
[condition]

set_opposite
Defines two input ports as logically opposite.

int set_opposite port1 port2

set_optimize_registers
Sets the optimize_registers attribute on the specified design or on the current design so that compile automatically invokes the Behavioral Compiler optimize_registers command to retime the design during optimization.

int set_optimize_registers [true | false]
[-design design_list]

set_output_delay
Sets output delay on pins or output ports relative to a clock signal.

int set_output_delay delay_value
[-clock clock_name
[-clock_fall][-level_sensitive]]
[-rise] [-fall] [-max] [-min] [-add_delay]
[-group_path group_name] port_pin_list
set_pad_type
Indicates the type of I/O pads needed on given ports.

int set_pad_type
[-example example_pad] [-exact exact_pad]
[-schmitt | -hysteresis]
[-pullup | -pulldown] [-opendrain]
[-opensource]
[vii vii] [-vih vih]
[vol vol] [-voh voh]
[vimin vmin] [-vimax vimax]
[vomin vomin] [-vomax vomax]
[-currentlevel currentlevel]
[-clock] [-no_clock]
[-slewrate slewratemaxvalue]
design_or_port_list

set_pipeline_stages
Sets directives to control the implementation of
DW03_mult_n_stage operators referenced by the
specified cells.

int set_pipeline_stages [cell_list]
[-min min_stages] [-fixed fixed_stages]
[-auto]

set_port_configuration
Provides the Shadow LogicDFT utility with
information about the input and output ports of the
specified elements that are to receive wrappers.

int set_port_configuration
-cell cell_design_ref_list
-port port_name [-tristate]
[-wrapper_exclude]
[-clock clock_name] [-read
signal_value_pin_pairs
[-write signal_value_pin_pairs

set_port_fanout_number
Sets the number of external fanout points driven by
specified ports in the current design.

int set_port_fanout_number
fanout_number port_list
set_port_is_pad
Sets the port_is_pad attribute on specified ports and/or designs to indicate that those ports are to have I/O pads attached.

    int set_port_is_pad [port_design_list]

set_port_location
Annotates the specified top-level port with X and Y coordinates to be used by FloorPlan Manager during execution of reoptimize_design.

    int set_port_location [-coordinate (x_coordinate y_coordinate)]
    port_name

set_prefer
Sets the preferred attribute on specified library gates.

    int set_prefer [-min] gate_list

set_propagated_clock
Specifies propagated clock latency.

    string set_propagated_clock object_list

set_register_type
Sets the latch_type or flip_flop_type attributes on designs or cell instances, to specify which sequential cells from the target library are to be used by compile.

    int set_register_type
    [[-exact] -latch example_latch]
    [[-exact] -flip_flop example_flip_flop]
    [cell_or_design_list]

set_resistance
Sets the resistance value on nets.

    int set_resistance value [-min] [-max]
    net_list
**set_resource_allocation**
Sets the resource_allocation attribute on the current design, thus specifying the type of resource allocation to be used by compile.

```c
int set_resource_allocation
none | area_only | constraint_driven
```

**set_resource_implementation**
Sets the resource_implementation attribute on the current design, thus specifying the type of resource implementation to be used by compile.

```c
int set_resource_implementation
area_only | constraint_driven | use_fastest
```

**set_rtl_load**
Sets an rtl load value for capacitance and resistance on pins, ports and nets.

```c
int set_rtl_load [-cap cvalue] [-res rvalue] [-min] [-max] pin_net_list
```

**set_scan_bidi**
Determines whether insert_scan and insert_dft configure specified bidirectional ports as inputs or outputs, or leave them untouched, during scan shift.

```c
int set_scan_bidi
bidir_mode -port port_list
```
**set_scan_configuration**

Specifies the scan chain design.

```plaintext
int set_scan_configuration
    [-add_lockup true | false]
    [-area_critical false]
    [-bidi_mode input | output | no_disabling ]
    [-chain_count integer_or_default]
    [-clock_gating entire_design | leaf_cell | superbuffer]
    [-clock_mixing no_mix | mix_edges | mix_clocks | mix_clocks_not_edges]
    [-create_test_clocks_by_system_clock_domain true | false ]
    [-dedicated_scan_ports true | false]
    [-disable true | false]
    [-existing_scan true | false]
    [-external_tristates disable_all | enable_one | no_disabling]
    [-hierarchical_isolation true | false]
    [-internal_clocks true | false ]
    [-internal_tristates disable_all | enable_one | no_disabling]
    [-methodology full_scan | partial_scan | none]
    [-multibit_segments true | false]
    [-physical true | false]
    [-prfile report_file_name]
    [-prtool cadence | avant]
    [-rebalance true | false]
    [-replace true | false]
    [-route true | false]
    [-route_signals all | global | serial | clocks | scan_enables]
    [-style multiplexed_flip_flop | clocked_scan | lssd | aux_clock_lssd | combinational | none]
    [-test_models true | false]
```

**set_scan_element**

Sets the scan_element attribute on specified design objects to determine whether or not insert_scan replaces them with scan cells.

```plaintext
int set_scan_element true | false
cell_design_ref_listf -multibit
multi-bit_list
```
**set_scan_link**
Declares a scan link for the current design.

```
int set_scan_link
scan_link_name wire | scan_out_lockup
```

**set_scan_path**
Specifies a scan chain for the current design.

```
int set_scan_path
scan_chain_name [ordered_list]
[-dedicated_scan_out true | false]
[-complete true | false]
```

**set_scan_register_type**
Specifies a list of scan sequential cells from the target library that are to be used by insert_scan or compile -scan when scan replacing designs or cell instances.

```
int set_scan_register_type [-exact]
-type example_scan_seq_cell_list
[cell_or_design_list]
```

**set_scan_segment**
Identifies logic in the current design that is to be designated a scan segment.

```
int set_scan_segment
scan_segment_name
[-access signal_type_pin_pairs]
[-contains member_list]
[-synthesizable true | false | default]
[-reverse_order true | false]
```

**set_scan_signal**
Specifies scan signals for the current design.

```
int set_scan_signal
scan_signal_type -port port_list
[-hookup pin [-sense sense]]
[-chain chain_list]
```
set_scan_style
Specifies the scan test implementation style for a design.

Note: Although this command is currently still supported, it and its associated scan synthesis command, insert_test, are outmoded and will become obsolete. Use set_scan_configuration -style, along with insert_scan, instead.

int set_scan_style style

set_scan_transparent
Sets the scan_latch_transparent attribute on specified design objects to determine whether or not level-sensitive sequential cells are modeled as transparent latches during automatic test pattern generation (ATPG).

int set_scan_transparent true | false
cell_design_ref_list
-multibit multi-bit_list -existing

set_scan_tristate
For specified tristate nets, determines whether insert_scan and insert_dft disable all drivers, enable exactly one driver, or leave the nets untouched during scan shift.

int set_scan_tristate disabling_option -net net_list

set_share_cse
Sets the share_cse attribute, which determines whether common subexpressions are shared during compile. By default, all common subexpressions are shared unless otherwise specified.

int set_share_cse [true | false]
[-design design_list]
list design_list
set_signal_type
Sets the signal type on a list of pins or ports.

```
int set_signal_type signal_type port_list
[-associated_clock clk]
[-index signal_index]
```

set_simple_compile_mode
Places Design Compiler into simple compile mode.

```
int set_simple_compile_mode [true | false]
[-verbose]
```

set_state_for_retiming
Sets the state_for_retiming attribute on cells in the current design. This command can effect both hierarchical cells and sequential leaf cells.

```
int set_state_for_retiming cell_list
preserve | dont_care
```

set_structure
Sets various structure attributes on a design or on a list of designs to determine whether and how the designs are structured during compile.

```
int set_structure [true | false]
[-design design_list]
[-boolean true | false]
[-boolean_effort low | medium | high]
[-timing true | false]
```

set_switching_activity
Sets (or resets) the toggle_rate and static_probability values for nets, pins, and ports of the current design.

```
int set_switching_activity
[-static_probability sp_value]
[-toggle_rate tr_value]
[-state_dep boolean_eq_of_pins]
[-path_dep sources_of_path]
[-transition_type rising | falling]
[-period period_value | -clock clock_name]
object_list
```
set_test_assume
Sets the test_assume attribute to a logic value to be assumed on specified cell output pins throughout test design rule checking, test pattern generation, and fault simulation.

int set_test_assume zero_or_one_value

set_test_dont_fault
Sets the test_dont_fault attribute on cells, pins, or ports, excluding the faults on these objects from test pattern generation, fault simulation, and fault coverage computation.

int set_test_dont_fault [-sa0] [-sal] object_list

set_test_hold
Sets the test_hold attribute to a logic value to be assumed on specified input ports during testing.

int set_test_hold zero_or_one_value

set_test_initial
Sets the test_initial attribute to a logic value to be assumed on specified cell output pins at the start of test design rule checking and fault simulation.

int set_test_initial zero_or_one_value

set_test_isolate
Sets the test_isolate attribute on the specified cells, pins, or ports, indicating that they are to be logically isolated and considered untestable during test design rule checking.

int set_test_isolate pin_cell_port_list
set_test_mask_fault
Masks faults on specified cells, pins, or ports for the next test pattern generation or fault simulation run.

```c
int set_test_mask_fault [-sa0] [-sa1] [-except except_object_list] object_list
```

set_test_methodology
Specifies the test implementation methodology for a design.

Note: Although this command is currently still supported, it and its associated scan synthesis command, insert_test, are outmoded and will become obsolete. Use set_scan_configuration -methodology along with insert_scan, instead.

```c
int set_test_methodology full_scan | partial_scan [-existing_scan]
```

set_test_require
Sets the test_require attribute to a logic value to be maintained on generated test vectors for specified pins.

```c
int set_test_require zero_or_one_value pin_list
```

set_test_signal
Specifies test-mode signals for the current design.

```c
int set_test_signal test_signal_type -port port_name
```

set_test_unmask_fault
Removes masks from previously masked faults on cells, pins, or ports for the next test pattern generation or fault simulation run.

```c
int set_test_unmask_fault [-sa0] [-sa1] -all | object_list
```
**set_timing_ranges**
Sets timing ranges for the current design.

```plaintext
int set_timing_ranges [timing_rangesf1]
[-library library_name]
```

**set_transform_for_retiming**
Sets the transform_for_retiming attribute on cells in the current design. This can effect both hierarchical cells and sequential leaf cells.

```plaintext
int set_transform_for_retiming cell_list
multiclass | decompose | dont_retime
```

**set_true_delay_case_analysis**
Sets the true_delay_case_analysis attribute, which specifies the input vector value to use for specified pins or ports of the current design for the -true and -justify options of report_timing.

```plaintext
int set_true_delay_case_analysis 0 | 1 | r | f | none port_pin_list
```

**set_ultra_optimization**
Sets the DC Ultra optimization mode and checks out the DC Ultra license, if available, for the current Design Compiler session.

```plaintext
int set_ultra_optimization [true | false]
[-force]
```

**set_unconnected**
Lists output ports to be unconnected.

```plaintext
int set_unconnected port_list
```

**set_ungroup**
Sets the ungroup attribute on specified designs, cells, or references, indicating that they are to be ungrouped during compile.

```plaintext
int set_ungroup object_list true | false
```
set_unix_variable
Sets the value of a UNIX environment variable.

string set_unix_variable variable_name
new_value

set_wire_load_min_block_size
Sets the wire load min_block_size attribute on the current design.

int set_wire_load_min_block_size size

set_wire_load_mode
Sets the wire_load_mode attribute on the current design, specifying how wire load models are to be used to calculate wire capacitance in nets.

int set_wire_load_mode mode_name

set_wire_load_model
Sets the wire_load_attach_name attribute on designs, ports, hierarchical cells of current design, or the specified cluster of the current design, for selecting a wire load model to use in calculating wire capacitance.

int set_wire_load_model -name model_name
[-library lib]
[-min] [-max] [-cluster cluster_name]
[object_list]

set_wire_load_selection_group
Specifies a selection group to use for determining a wire load model to be assigned to designs and cells or to a specified cluster. This command is supported only for the enclosed wire load mode.

int set_wire_load_selection_group
[-library lib] [-min]
[-max] [-cluster cluster_name] group_name
[object_list]
set_wired_logic_disable
Sets the wired_logic_disable attribute on the specified
ECL designs to indicate whether or not the creation of
wired OR logic is to be disabled when optimizing the
designs using compile.

int set_wired_logic_disable object_list
   [true | false]

set_wrapper_element
Sets the wrapper_element attribute on a list of
elements around which preview_dft and insert_dft are
to insert a wrapper when the Shadow LogicDFT
utility is enabled.

int set_wrapper_element cell_design_ref_list
   -type wrapper_type

setenv (dctcl-mode only)
Built-in Tcl command.

sh
Sends a command to the UNIX operating system.

int sh command

simplify_constants
Propagates constants and other information in the
current design.

int simplify_constants
   [-verify] [-verify_hierarchically]
   [-verify_effort low | medium | high]
   [-boundary_optimization]

sizeof_collection
Returns the number of objects in a collection. For use
in dc_shell-t (Tcl mode of dc_shell) only.

int sizeof_collection name

socket (dctcl-mode only)
Built-in Tcl command.
sort_collection
Sorts a collection based on one or more attributes, resulting in a new, sorted collection. The base collection remains unchanged. For use in dc_shell-t (Tcl mode of dc_shell) only.

```
string sort_collection [-descending] collection criteria
```

source (dctl-mode only)
Built-in Tcl command.

split (dctl-mode only)
Built-in Tcl command.

string (dctl-mode only)
Built-in Tcl command.

sub_designs_of
Gets the subdesigns according to the options.

```
sub_designs_of
[-hierarchy]
[-in_partition | -partition_only]
[-dt_only | -ndt_only]
[-multiple_instances | -single_instances] design
```

sub_instances_of
Gets the subinstances according to the options.

```
sub_instances_of
[-hierarchy]
[-in_partition] [-partition_only]
[-dt_only] [-ndt_only]
[-of_references reference_list]
[-master_instance] design
```

subst (dctl-mode only)
Built-in Tcl command.

suppress_message (dctl-mode only)
Built-in Tcl command.


**switch** (dctcl-mode only)

Built-in Tcl command.

**syntax_check**

Enables or disables Syntax Checker syntax_check mode, which checks commands for syntax errors without executing them.

```tcl
int syntax_check true | false
```

**tcl::history** (dctcl-mode only)

Built-in Tcl command.

**tclLog** (dctcl-mode only)

Built-in Tcl command.

**tell** (dctcl-mode only)

Built-in Tcl command.

**time** (dctcl-mode only)

Built-in Tcl command.

**trace** (dctcl-mode only)

Built-in Tcl command.

**trace_nets**

Enables global net tracing during check_test on the specified nets in the current design.

```tcl
int trace_nets
hierarchical_net_list
```

**transform_csa**

Transforms arithmetic operators (for example, addition, subtraction, and multiplication) into the carry save adder (CSA) operator, for the specified design or for the current design.

```tcl
int transform_csa [design_name] [-labelled_only]
[-decompose_multiply all | none | constant]
[-duplicate] [-area] [-group] [-dont_split]
```
translate
Translates a design from one technology to another.

```
int translate [-verify]
[-verify_hierarchically]
[-verify_effort low | medium | high]
```

unalias
Removes alias definitions.

```
list unalias [-all] [alias_list]
```

ungroup
Removes a level of hierarchy.

```
int ungroup cell_list | -all
[-prefix prefix_name]
[-flatten] [-simple_names] [-soft]
```

uniquify
Removes multiply-instantiated hierarchy in the current design by creating a unique design for each cell instance.

```
int uniquify [-force]
[-base_name base_name]
[-cell cell_list]
[-reference design_name]
[-new_name new_design_name]
```

unschedule
Permits Behavioral Compiler to reschedule I/O operations into cycles different from those defined in the original HDL description.

```
int unschedule operation_names
```

unset (dctcl-mode only)
Built-in Tcl command.

unsuppress_message (dctcl-mode only)
Built-in Tcl command.
**untrace_nets**
Disables global net tracing during check_test on any specified nets for which net tracing had been previously enabled by trace_nets.

```plaintext
int untrace_nets
hierarchical_net_list | -all
```

**update (dctcl-mode only)**
Built-in Tcl command.

**update_clusters**
Updates the clusters associated with the current design to reflect the changes made to a subdesign.

```plaintext
int update_clusters cell_list
```

**update_lib**
Reads in a specified library file and uses it to update an existing technology, synthetic, or symbol library.

```plaintext
int update_lib [-overwrite] [-permanent] library_name file_name [-no_warnings]
```

**update_script**
Modifies an old dc_shell script to use current dc_shell commands.

```plaintext
int update_script [-from_version version] script_file_name [-output_file output_file_name]
```

**update_timing**
Updates timing information on the current design.

```plaintext
int update_timing
```

**uplevel (dctcl-mode only)**
Built-in Tcl command.

**upvar (dctcl-mode only)**
Built-in Tcl command.
variable (dctcl-mode only)
   Built-in Tcl command.

vwait (dctcl-mode only)
   Built-in Tcl command.

which
   Displays the pathname of one or more files in dc_shell or in dc_shell-t (Tcl mode of dc_shell).
   list which file_names

while
   Loop execution control structure.
   while { expression } {
     loop-statement-block
   }

write
   Writes a design netlist or schematic from dc_shell to a file.
   int write [-format output_format] 
               [-hierarchy] [-no_implicit] [-modified] 
               [-output output_file_name] 
               [-library library_name] 
               [design_list] 
               [-names_file name_mapping_files] 
               [-donot_expand_dw] 
               [-rtl_script script_name] 
               [-rules_name rules_name]

write_bsd_protocol
   Writes a boundary-scan protocol file.
   int write_bsd_protocol [-out protocol_file] [-format tpf | stil]

write_bsd1
   Generates the boundary-scan description language (BSDL) file for a boundary-scan design.
   int write_bsd1 [-naming_check VHDL | BSDL | none] [-output file_name] 
                  [-effort low | medium | high]
**write_clusters**

Writes to a file in Physical Design Exchange Format (PDEF) the physical cluster annotations associated with a design.

```bash
int write_clusters [-design design_name] [-output new_cluster_file_name] [-no_attributes] [-hier_cells] [-new_cells_only original_cluster_file_name]
```

**write_compare_design_script**

Saves the compare_design script, which contains dc_shell commands to be used during verification with balance_buffer, compare_design, compile, insert_pads, reoptimize_design, replace_fpga, and translate.

```bash
int write_compare_design_script
```

**write_compile_script**

Writes a compile script for the specified design.

```bash
write_compile_script [-absolute_paths] [-hierarchy] [-format dcsh | dctcl] [-no_reports] -destination pass [design]
```
write_constraints

Writes constraints for the place and route tools.

int write_constraints [-output file_name]
[-format synopsys | sdf | sdf-v2.1]
[-max_paths max_path_number]
[-nworst nworst_number]
[-max_path_slack slack_value]
[-cover_design | -cover_nets]
[-net_priorities]
[-min_net_priority min_priority_number]
[-max_net_priority max_priority_number]
[-low_priorities] [-max_path_timing]
[-net_timing] [-load_delay net | cell]
[-net_capacitance] [-subtract_pin_cap]
[-cell_groups]
[-hierarchy] [-by_input_pin_name]
[-by_output_pin_name]
[-max_nets max_net_number]
[-from start_point_list]
[-to end_point_list]
[-through through_point_list]

write_design_lib_paths

Writes to a file the paths to which design libraries are mapped.

int write_design_lib_paths [-filename file_name] [-dc_setup]

write_designlist

Writes a list of designs referenced by the specified design or by the current design.

int write_designlist [-output listfile]
[design]
write_environment

Writes the variable settings and constraints for the specified cells or designs.

write_environment

[{-cells cell_list} | {-designs design_list}]
[-format dcsh | dctcl]
[-output file_name]
[-suffix suffix]
[-environment_only]
[-constraints_only]
[-no_lib_info]

write_file (dctcl-mode only)

Built-in Tcl command.

write_layout_scan

For the current design, writes scan chain information for performing scan chain reordering using third-party place and route tools.

int write_layout_scan
[-out output_command_file] [-noclockdomain]

write_lib

Writes a compiled library to disk in Synopsys database, EDIF, or VHDL format.

int write_lib library_name [-format db | edif | vhdl] [-output file_name]
[-names_file file_list]

write_makefile

Writes a makefile that defines the dependencies and commands required to compile the specified design.

write_makefile

[-absolute_paths]
[-dependencies depends]
[-dc_shell exec_name]
[-format dcsh | dctcl]
-destination pass
[-target target_name]
[-lsf [-bsubargs bsub_args]]
[design]
write_parasitics

Writes parasitics in SPEF format to a disk file for the
delay calculation tools.

int write_parasitics [-output file_name]
[-ratio ratio_number] [-script]

write_partition

Writes the database for a design into the ACS data
structure.

write_partition
-type pre | post
[-destination pass]
[-hierarchy]
[design]

write_partition_constraints

Writes out the timing constraints for a design.

write_partition_constraints
[-hierarchy]
[-format dcsh | dctcl]
-destination pass
[design]

write_power

Calculates and saves dynamic and static power
information of a design or instance for interface with
HDL Advisor.

int write_power [-net] [ -cell]
[-only cell_or_net_list]
[-cumulative] [-flat]
[-exclude_boundary_nets]
[-analysis_effort low | medium | high]

write_rtl_load

Writes rtl load commands for the current design to a
script.

int write_rtl_load [-format dctcl | dcsh]
[-output file_name]
write_script
Writes dc_shell commands to save the current
settings.

int write_script [-hierarchy]
[-no_annotated_check] [-no_annotated_delay]
[-full_path_lib_names]
[-format dctcl | dcsh]
[-output file_name]

write_sdc
Writes out a script in Synopsys Design Constraints
(SDC) Version 1.1 format.

int write_sdc file_name

write_sdf
Writes a Standard Delay Format (SDF)
back-annotation file.

string write_sdf [-version sdf_version]
[-instance inst_name]
file_name

write_test
Formats the test patterns for the current design into
one or more test vector files.

int write_test [-input test_program_name]
[[-output output_vector_file_name] | [-cumulative]]
[[-format test_program_format]
[[-first n_patterns] [-parallel]
[[-part_number part_number]
[[-revision revision]

write_test_protocol
Writes a test protocol file.

int write_test_protocol [-out file_name]
[-format tpf | stil]

write_testsim_lib
Note: TestSim is obsolete with 1999.10 and has been
replaced by the TetraMax fault simulator. For more
information, see the TetraMax documentation.
write_timing

Writes leaf cell pin-to-pin timing information to a disk file.

int write_timing [-output timing_file_name]
[-load_delay net | cell]
[-design design_name]
Commands Specific to dcsh Mode

The following commands are available only in dcsh mode:

- allocate_budgets
- check_unmapped
- execute
- include
- remove_variable
- write_designlist

Commands Specific to dctcl Mode

The following commands are available only in dctcl mode:

- acs_check_directories
- acs_compile_design
- acs_create_directories
- acs_get_path
- acs_recompile_design
- acs_refine_design
- acs_report_directories
- add_to_collection
- after
- allocate_partition_budgets
- append
- apropos
- array
- auto_execok
- auto_import
- auto_load
- auto_load_index
- auto_qualify
- begin_incr_mode
- binary
- catch
- clock
- close
- compare_collections
- compile_partitions
- concat
- copy_collection
- create_command_group
- create_pass_directories
- current_design_name

124 Synthesis Commands
date
define_proc_attributes
encoding
eof
error
error_info
eval
exec
expr
fblocked
fconfigure
fcopy
file
fileevent
filter_collection
flush
for
foreach_in_collection
format
get_cells
get_clocks
get_clusters
get_designs
get_generated_clocks
get_lib_cells
get_lib_pins
get_libs
get_multibits
get_nets
get_object_name
get_path_groups
get_pins
get_ports
get_references
get_timing_paths
getenv
gets
glob
global
incr
index_collection
info
interp
is_false
is_true
join
lappend
lindex
linsert
list_attributes
list_files
list_licenses
llength
lminus
lrange
lreplace
ls
lsearch
lsort
man
namespace
open
package
parse_proc_arguments
pid
print_suppressed_messages
print_variable_group
printenv
printvar
proc
proc_args
proc_body
puts
query_objects
read_db
read_edif
read_partition
read_sdc
read_verilog
read_vhdl
redirect
regexp
regsub
remove_from_collection
remove_pass_directories
report_pass_data
return
scan
seek
set
set_compile_partitions
setenv
sizeof_collection
socket
sort_collection
source
split
string
sub_designs_of
sub_instances_of
subst
suppress_message
switch
tcl::history
tclLog
tell
time
trace
unset
unsuppress_message
update
uplevel
upvar
variable
vwait
write_compile_script
write_makefile
write_partition
write_partition_constraints
Synthesis Variables

The synthesis tools define a set of variables that are used to control its behavior.

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

Default value for this variable is “area”.

acs_bs_exec
For use in dc_shell-t (Tcl mode of dc_shell) only. Specifies the location of the budget_shell executable; the default is shown under SYNTAX. This variable is used by the acs_refine_design and acs_recompile_design commands to run design budgeting.

Default value for this variable is “$SYNOPSYS/sparcOS5/syn/bin/budget_shell”.

acs_budget_output_file_suffix
For use in dc_shell-t (Tcl mode of dc_shell) only. Specifies the default suffix for log files generated by the allocate_partition_budgets command. The default is btcl.out.

Default value for this variable is “btcl.out”.

acs_budget_script_file_suffix
For use in dc_shell-t (Tcl mode of dc_shell) only. Specifies the default suffix for design budgeting script files generated by the allocate_partition_budgets command; the default is btcl.

Default value for this variable is “btcl”.
**acs_budgeted_cstr_suffix**
For use in dc_shell-t (Tcl mode of dc_shell) only.
Specifies the suffix for constraint files generated by the derive_partition_budgets command; the default is con.
Default value for this variable is “con”.

**acs_compile_script_suffix**
For use in dc_shell-t (Tcl mode of dc_shell) only.
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. The default is autoscr.
Default value for this variable is “autoscr”.

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

**acs_cstr_report_suffix**
For use in dc_shell-t (Tcl mode of dc_shell) only.
Specifies the default suffix for constraint reports generated during the automated compile process; the default is cstr.
Default value for this variable is “cstr”.

**acs_db_suffix**
For use in dc_shell-t (Tcl mode of dc_shell) only.
Specifies the default suffix for .db files that are read or written during the automated compile process. The default is db.
Default value for this variable is “db”.

__Synthesis Variables__
129
acs_dc_exec
For use in dc_shell-t (Tcl mode of dc_shell) only.
Specifies the location of the dc_shell executable. This variable is used by the acs_compile_design,
acs_refine_design, and acs_recompile_design commands to generate the makefile. The default is shown under SYNTAX.
Default value for this variable is “$SYNOPSYS/sparcOS5/syn/bin/dc_shell”.

acs_global_user_compile_strategy_script
For use in dc_shell-t (Tcl mode of dc_shell) only.
Specifies the filename for the user-defined default compile strategy; the compile strategy is a script that includes commands to set the compile variables, set the compile attributes, and run the compile command.
The default value for this variable is “default compile”.

acs_log_file_suffix
For use in dc_shell-t (Tcl mode of dc_shell) only.
Specifies the default suffix for log files generated during the automated compile process; the default is log.
Default value for this variable is “log”.

acs_makefile_name
For use in dc_shell-t (Tcl mode of dc_shell) only.
Specifies the filename for the makefile generated by the write_makefile command and run by the compile_partitions command. The default is Makefile.
Default value for this variable is “Makefile”.

130Synthesis Variables
acs_num_parallel_jobs
For use in dc_shell-t (Tcl mode of dc_shell) only. Specifies the number of compile jobs to run in parallel when using gmake as the make utility; the default is 1. Because the Load Sharing Facility (LSF) bsub command automatically determines the number of parallel compile jobs to run based on available resources, this variable is not used when acs_use_lsf is true.

Default value for this variable is “1”.

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

Default value for this variable is “scr”.

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

Default value for this variable is “qor”.

acs_script_mode
For use in dc_shell-t (Tcl mode of dc_shell) only. Specifies the dc_shell mode used by Automated Chip Synthesis for the compile process when running one of the pass commands (acs_compile_design, acs_refine_design, or acs_recompile_design). The generated constraint files, script files, and makefile reflect this format selection. Valid values for this variable are dcsh (the default) or dctcl.

Default value for this variable is “dcsh”.
acs_timing_report_suffix
For use in dc_shell-t (Tcl mode of dc_shell) only. Specifies the suffix for timing reports generated during the automated compile process; the default is tim.

Default value for this variable is “tim”.

acs_tr_exec
For use in dc_shell-t (Tcl mode of dc_shell) only. Specifies the location of the transcript executable; the default is shown under SYNTAX. This variable is used by the acs_refine_design and acs_recompile_design commands to convert generated constraints to dcsh format.

Default value for this variable is “$SYNOPSYS/sparcOS5/syn/bin/dc-transcript”.

acs_use_lsf
For use in dc_shell-t (Tcl mode of dc_shell) only. Specifies which make utility you want to use to process the compile job. When false (the default), Automated Chip Synthesis uses gmake as the make utility. If you want to use the Load Sharing Facility (LSF) bsub command, set this variable to true.

Default value for this variable is “false”.

acs_user_budgeting_script
For use in dc_shell-t (Tcl mode of dc_shell) only. Specifies the file name for the user-defined budgeting script; the default is budget.scr.

Default value for this variable is “budget.scr”.
acs_user_compile_strategy_script_suffix
For use in dc_shell-t (Tcl mode of dc_shell) only.
Specifies the suffix for user-defined partition compile strategies; the default is compile. A compile strategy is a script that includes commands to set the compile variables, set the compile attributes, and run the compile command.

Default value for this variable is “compile”.

acs_work_dir
For use in dc_shell-t (Tcl mode of dc_shell) only.
Specifies the root of the ACS project directory. During startup, this variable is set to the current working directory. This value is used when locating input files for ACS commands and when generating absolute path names.

Default value for this variable is “[pwd]”.

atpg_bidirect_output_only
When true, all bidirectionals in the circuit are forced in the output mode during test generation. The default is false. Setting this variable to true can affect the results of ATPG with respect to fault coverage and run time.

Default value for this variable is “true”.

auto_link_disable
When true, specifies that the code to perform an
auto_link during any Design Compiler command
should be disabled, resulting in a speedup in
command processing. This speedup is important in
backannotation commands like set_load,
set_resistance, and set_annotated_delay where
potentially thousands of such commands are executed
in sequence. Disabling the auto_link code can
significantly improve the speed with which such
commands get executed.
Default value for this variable is “false”.

auto_link_options
Specifies the link command options to be used when
link is invoked automatically by various Design
Compiler and Test Compiler commands (for example,
create_schematic and compile). The default is -all. To
find the available options, refer to the link command
manual page.
Default value for this variable is “-all”.

auto_wire_load_selection
When true, turns on the automatic selection of the
wire load model 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.
Default value for this variable is “true”.

134 Synthesis Variables
**bc_allow_shared_memories**
For Behavioral Compiler, enables sharing of memories between processes.

Default value for this variable is “false”.

**bc_chain_read_into_mem**
Enables Behavioral Compiler to chain input reads directly into a memory even if the inputs to the memory have been specified as non-chainable.

When true (the default), Behavioral Compiler chains input reads into a memory even if the memory inputs are declared as non-chainable by having stable = false specified on the memory sldb file.

Default value for this variable is “true”.

**bc_chain_read_into_oper**
Enables Behavioral Compiler to chain input reads directly into an operation even if the inputs to the operation have been specified as non-chainable.

When true (the default), Behavioral Compiler chains input reads into an operation even if the operation inputs have been specified as non-chainable using the dont_chain_operations -into command.

Default value for this variable is “true”.

**bc_constrain_signal_memories**
For Behavioral Compiler, determines whether shared memory accesses are scheduled as signals or variables.

Default value for this variable is “false”.

**bc_detect_array_accesses**

For Behavioral Compiler, determines whether precedences (dependency-related constraints) between non-conflicting pairs of array accesses to the same register file are automatically ignored. Arrays are mapped to register files if bc_use_registerfiles is set to true before analyzing and elaborating the behavioral source code.

Default value for this variable is “false”.

**bc_detect_memory_accesses**

For Behavioral Compiler, determines whether precedences (dependency related constraints) between non-conflicting pairs of memory accesses are automatically ignored.

Default value for this variable is “false”.

**bc_enable_analysis_info**

Enables the generation of BCView analysis information. When true, BCView analysis information is created for designs processed by subsequent Behavioral Compiler commands. In BCView, analysis information must be created for the designs in order to view the links between scheduled behavioral designs and HDL code. This analysis information is created when designs are processed by the following Behavioral Compiler commands: analyze, elaborate -s, bc_check_design, and schedule.

Default value for this variable is “false”.
**bc_enable_chaining**

For Behavioral Compiler, enables chaining of synthetic library operations. When true (the default), schedule chains two or more cascaded synthetic library operations in the same cycle if the propagation delay through these operations is less than the clock period. When false, schedule does not chain cascaded operations, even if their propagation delay is less than the clock period.

Default value for this variable is “true”.

**bc_enable_multi_cycle**

For Behavioral Compiler, enables inference of multi-cycle synthetic library operations. When true (the default), schedule makes synthetic library operations multi-cycle if their propagation delays are greater than the clock period. When false, schedule does not make operations multi-cycle, even if their propagation delays are greater than the clock period.

Default value for this variable is “true”.

**bc_enable_speculative_execution**

When true, enables speculative execution for Behavioral Compiler, where data operations are precalculated before it is known whether or not the results are needed. When false (the default), speculative execution is disabled.

Default value for this variable is “false”.

**bc_estimate_mux_input**

For Behavioral Compiler, specifies the number of mux inputs for bc_time_design to use when estimating the mux delay timing, enabled by bc_estimate_mux_delay.

Default value for this variable is “4”.
**bc_estimate_timing_effort**
For Behavioral Compiler, specifies the level of timing estimation effort to be used in addition to the operator timing performed by bc_time_design. Allowed values are zero, low, medium, and high (the default).

Default value for this variable is “high”.

**bc_fsm_coding_style**
Controls the state assignment for the controller generated by Behavioral Compiler. Allowed values are one_hot (the default), two_hot, or binary. A one-hot coded state machine contains one register for every state in the controller. This results in fast logic, but uses more registers than the other options. A binary coded state machine uses a minimum number of registers, potentially at the expense of increased cycle time. The two-hot coding is a compromise between the two methods, with an intermediate number of registers and an intermediate propagation delay.

Default value for this variable is “one_hot”.

**bc_group_eql_logic**
When true (the default), Behavioral Compiler groups all logic gates for the equal/non-equal operation. This grouped logic will be ungrouped during compile unless bc_dont_ungroup is specified.

Default value for this variable is “true”.

**bc_group_index_logic**
When true (the default), Behavioral Compiler groups all logic gates for the array indexing operation. This grouped logic will be ungrouped during compile unless bc_dont_ungroup is specified.

Default value for this variable is “true”.

138 Synthesis Variables
**bc_infer_multibit**

Enables Behavioral Compiler to infer multibit registers if they are available in the target technology library.

When true, causes Behavioral Compiler to use multibit registers in the design if multibit registers are available in your technology library.

Default value for this variable is “=“.

**bc_minimum_multibit_component_width**

Specifies the minimum register size for Behavioral Compiler to use when building a multibit register.

The value of this variable represents the minimum bit width for Behavioral Compiler to use when building a multibit register; the default is 4. Behavioral Compiler is constrained to multibit registers that have a bit width greater than or equal to the value specified.

Default value for this variable is “false”.

**bc_report_filter**

For Behavioral Compiler, to be used only with report_schedule -op and -var. Contains a string to be used to filter the operators and variables for report_schedule -op and report_schedule -var. Resource types are tested for the specified string; only those that contain the string are reported on.

Default value for this variable is the empty string ““”. 
**bc_synrtl_map_to_gtech**

When true (the default), Behavioral Compiler uses the GTECH technology library when carrying out the preprocessing step before generating a synthesizable RTLOUT. This speeds up the sharing process and, more importantly, makes the synthesizable RTLOUT technology-independent. When false, Behavioral Compiler uses the real target library.

Default value for this variable is “true”.

**bc_synrtl_write_dcsh_and_dctcl**

When false (the default), the write -rtl_script command generates one copy of the constraint.script file in a format that depends on the current mode in which dc_shell is running. That is, if dc_shell is running in dcsh mode, the file is in dcsh format, and if dc_shell is running in dctcl mode, the file is in dctcl format. When this variable is set to true, regardless of the mode in which dc_shell is running, write generates two copies of the constraint.script file: one in dcsh format, and one in dctcl format.

Default value for this variable is “false”.

---

140 Synthesis Variables
**bc_synrtl_write_precompiled_designware**

When false (the default), the write command does not write out precompiled DesignWare components to the synthesizable RTLOUT. The DesignWare parts are instantiated in the synthesizable RTLOUT through linking; therefore, you must provide a .db file in a directory in the search_path, or linking will fail. Accepting the default value of false is recommended, because linking takes less effort than rebuilding. However, you must ensure that the design environment and constraints are consistent with the core (other) design to achieve a consistent and meaningful quality of the resulting design.

Default value for this variable is “false”.

**bc_synrtl_write_preserved_functions**

When false (the default), the write command does not write out preserved functions to the synthesizable RTLOUT. Preserved functions are assumed to be compiled. you must provide a .db file in a directory in the search_path, or linking will fail. Accepting the default value of false is recommended, because linking takes less effort than rebuilding. However, you must ensure that the design environment and constraints are consistent with the core (other) design to achieve a consistent quality of result.

Default value for this variable is “false”.
**bc_time_all_sequential_op_bindings**
For Behavioral Compiler, controls the delay calculation for operations mapped onto DW03_mult_n_stage. When true, bc_time_design times all potential implementations (that is, modules for which a binding exists) for DW03_mult_n_stage. When false (the default), bc_time_design chooses the implementation that has the maximum number of pipeline stages for timing.

Default value for this variable is “false”.

**bc_use_registerfiles**
When true, indicates to use the array style in which array accesses are treated as ARR_READ and ARR_WRITE operators. When false (the default), indicates to use the array style in which an array is modeled as a register (one-dimensional value).

Default value for this variable is “false”.

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

Default value for this variable is ‘”]“.

**bus_extraction_style**
This variable affects the read command with the edif format option. It specifies the style used to extract the base name of net arrays, port arrays, and cell instance arrays in EDIF files. This variable is used with bus_naming_style and bus_dimension_separator_style to name bus members created in Design Compiler.

Default value for this variable is “%s[%d:%d]”. 
**bus_inference_descending_sort**
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.

Default value for this variable is “true”.

**bus_inference_style**
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 by this variable), the individual bits can be inferred into a port bus. If you specify an invalid value, no port busses are inferred.

Default value for this variable is “”.

**bus_minus_style**
This variable affects the read command with the VHDL format option. For this option, this variable controls the naming of individual members of bit-blasted port, instance, or net busses with negative indices.

Default value for this variable is “-%d”.

---

Synthesis Variables 143
**bus_multiple_separator_style**
This variable affects the naming of multibit cells during multibit mapping. This variable is used to name a multibit cell which implements bits that do not form a range. The default is used if an invalid value is specified.

Default value for this variable is “,”.

**bus_naming_style**
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 bussing option.

Default value for this variable is “%s[%d]”.

**bus_range_separator_style**
This variable affects the write command with the EDIF format option and the create_schematic command with the bussing option.

Default value for this variable is “:”.

**cache_dir_chmod_octal**
Cache directories are created with their mode bits set to the value of cache_dir_chmod_octal. The value of this variable is a string which is translated to an octal number. There are separate variables for directories and files to allow the sticky bit to be set.

Default value for this variable is “1777”.

---

144 Synthesis Variables
**cache_file_chmod_octal**

Cache files are created with their mode bits set to the value of cache_file_chmod_octal. The value of this variable is a string which is translated to an octal number. Cache directories use a different variable to set there mode bits (cache_dir_chmod_octal). There are separate variables for directories and files to allow the sticky bit to be set.

Default value for this variable is “664”.

**cache_read**

This variable specifies a list of directories. Each directory can contain a cache that will be read from wherever a cache entry is needed. If the variable is set to an empty list { }, then there are no directories containing caches and cacheing is effectively turned off. A Synopsys cache is stored as a UNIX directory tree rooted in one of the directories listed in the cache_read variable.

Default value for this variable is “{ }”.

**cache_read_info**

When true, an informational message will be printed each time a cache element is read. The default is false.

Default value for this variable is “false”.

**cache_write**

This variable specifies the directory where optimized and unoptimized synlib parts will be written, if they are not already in the cache (see cache_read(3)). If the variable is set to an empty string (), then synlib parts will not be written to the cache. A Synopsys cache is stored as a UNIX directory tree rooted in the directory listed in the cache_write variable.

Default value for this variable is “”.
**cache_write_info**

If true, an information message will be printed each time a cache element is written.

Default value for this variable is “false”.

**change_names_dont_change_bus_members**

This variable is for the change_names command, and affects bus members only of bussed 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.

Default value for this variable is “false”.

**change_names_update_inst_tree**

When true, enables the change_names command to update the instance tree(s) for all designs in dc_shell whenever names change, so that objects' instance-specific attributes are not lost. Notice that it updates the instance trees of all designs in dc_shell, not just the current design.

Default value for this variable is “false”.

**check_error_list**

For use in both dcsh and dctcl modes. Specifies a list of error codes for which messages are to be checked during the current Design Analyzer/dc_shell session; by default, no error message is checked.

Default value for this variable is “{}”.
**collection_result_display_limit**

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

Default value for this variable is “100”.

**command_log_file**

Specifies the name of the file to which a log of the initial values of variables and commands executed is written. If the value is an empty string, a command log file is not created.

Default value for this variable is “./command.log”.

**company**

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

Default value for this variable is “”.  

**compatibility_version**

Sets the default behavior of the system to be the same as the Synopsys software version specified in the variable. This provides compatibility for script command files written in previous software versions. The scripts are actually 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.

Default value for this variable is “”. 

---

Synthesis Variables 147
**compile_assume_fully_decoded_three_state_busss**

When true, compile and translate assume that three-state busses are fully decoded and, therefore, can be replaced by multiplexed busses when mapping to a library that contains no three-state cells. Default is false.

Default value for this variable is “false”.

**compile_automatic_clock_phase_inference**

When set to strict, compile will attempt to determine the desired clock phase for each unmapped register, and will not allow opposite phase devices to be used in constructing registers. When set to relaxed, compile will allow 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 will ignore clock phase during sequential mapping. The default is strict.

Default value for this variable is “strict”.

**compile_checkpoint_cpu_interval**

Specifies a time, in minutes, to be used as the interval between each automatic checkpoint. The default value, 0.0, indicates that automatic checkpointing is not enabled. The default filename is ./CHECKPOINT.db. The filename can be changed using the variable compile_checkpoint_filename.

Default value for this variable is “0.0”.

---

148 Synthesis Variables
**compile_checkpoint_filename**
Specifies the name of the file to which the database containing all hierarchy of the checkpointed design is to be written. The filename can have an absolute or relative path, but the specified directory must already exist and be writable.

Default value for this variable is “./CHECKPOINT.db”.

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

Default value for this variable is “false”.

**compile_checkpoint_pre_area_filename**
Specifies the name of the file to which the db containing all hierarchy of checkpointed design is written before Area-Recovery phase. If the value is none, a checkpointed db file is not created. The filename can have an absolute or relative path, but the specified directory must already exist and be writable.

Default value for this variable is “./CHECKPOINT_PRE_AREA.db”.

**compile_checkpoint_pre_delay_filename**
Specifies the name of the file to which the database containing all hierarchy of the checkpointed design is to be written before the delay optimization phase. If the value is none, a checkpointed db file is not created. The filename can have an absolute or relative path, but the specified directory must already exist and be writable.

Default value for this variable is “./CHECKPOINT_PRE_DELAY.db”.
compile_checkpoint_pre_drc1_filename
Specifies the name of the file to which the database containing all hierarchy of the checkpointed design is written before Design Rule Fixing Phase 1. If the value is none, a checkpointed db file is not created. The filename can have an absolute or relative path, but the specified directory must already exist and be writable.

Default value for this variable is “/CHECKPOINT_PRE_DRC1.db”.

compile_checkpoint_pre_drc2_filename
Specifies the name of the file to which the database containing all hierarchy of the checkpointed design is written before Design Rule Fixing Phase 2. If the value is none, a checkpointed db file is not created. The filename can have an absolute or relative path, but the specified directory must already exist and be writable.

Default value for this variable is “/CHECKPOINT_PRE_DRC2.db”.

compile_cpu_limit
Specifies a time, in minutes, to be used as the limit for the amount of time to be spent in the phases after structuring and mapping. Optimization aborts when the limit is reached. The default value, 0.0, indicates that there is no limit.

Default value for this variable is “0.0”.

compile_create_mux_op_hierarchy
When true (the default), compile creates all MUX_OP implementations with their own level of hierarchy. When false, compile removes this level of hierarchy.

Default value for this variable is “true”.

150 Synthesis Variables
**compile_create_wire_load_table**
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).

Default value for this variable is “false”.

**compile_delete_unloaded_sequential_cells**
A design can contain sequential cells that drive no loads. During compile, the logic driven by a sequential cell might be optimized away, resulting in an inferred no-load, or no path to any primary output. By default, compile deletes such sequential cells. To retain such cells, set variable compile_delete_unloaded_sequential_cells to false.

Default value for this variable is “true”.

**compile_disable_hierarchical_inverter_opt**
This variable affects the behavior of the boundary-optimization feature in the compile command. By default, boundary-optimization will try to push inverters across hierarchy 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 that could have improved the design.

Default value for this variable is “false”.

---
compile_dont_touch_annotated_cell_during_inplace_opt
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.
Default value for this variable is “false”.

compile_dont_use_dedicated_scanout
When 1 (the default), test-ready compile (compile -scan), and subsequent compiles, do not use a scan cell’s dedicated scan-out pin for functional connections. Instead, compile -scan connects the dedicated scan-out pin to the scan-in pin of the same scan cell to form a loop that models the timing and load impact of routing the scan chain.
Default value for this variable is 1.

compile_dw_simple_mode
This variable is for use only with the set_simple_compile_mode command.
Default value for this variable is “false”.

compile_fix_cell_degradation
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.
Default value for this variable is “false”.

152 Synthesis Variables
**compile_hold_reduce_cell_count**
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.

Default value for this variable is “false”.

**compile_ignore_wire_area**
When true, compile ignores wire area for wire delay calculations. When false (the default), compile uses the wire area in the wire delay calculations.

Default value for this variable is “false”.

**compile_implementation_selection**
When true (the default), compile re-evaluates the current implementation of a synthetic library module and replaces it if appropriate for optimizing the design. When false, this optimization is disabled, saving CPU time with a potential loss of quality of results.

Default value for this variable is “true”.

**compile_instance_name_prefix**
Specifies the prefix used in generating cell instance names when compile is executed.

Default value for this variable is “U”.

**compile_instance_name_suffix**
Specifies the suffix used for generating cell instance names when compile is executed.

Default value for this variable is “”. 

---

Synthesis Variables 153
**compile_log_format**
Controls the format of the columns to be displayed during the mapping phases of compile and reoptimize_design. 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, the `%area` AREA column, and so on.

Default value for this variable is “%elap_time %area %wns %tns %drc %endpoint”.

**compile_mux_no_boundary_optimization**
When true, compile does not perform boundary optimization on MUX_OP implementations. When false (the default), compile performs boundary optimization on all MUX_OP implementations. MUX_OP are created by HDL Compiler when the infer_mux attribute is set to a full case statement in the HDL code. For each full case statement with infer_mux, a generic MUX_OP design is created. This design is hierarchical, and Design Compiler maps the MUX_OP to a tree of multiplexers.

Default value for this variable is “false”.

**compile_negative_logic_methodology**
When true, compile and translate connect floating inputs to logic 1. The default value (false) causes floating inputs to be connected to logic 0.

Default value for this variable is “false”.
**compile_new_boolean_structure**
When true, turns on the new Boolean (non-algebraic) structure optimization in the structuring phase of compile if the structure_boolean attribute is set. There are three effort levels which are based on the relative CPU time to be spent in Boolean structure optimization. If compile_new_boolean_structure is false, Boolean structure optimization in a version earlier than 1997.07 is used. The default is false.

Default value for this variable is “false”.

**compile_no_new_cells_at_top_level**
When true, no new cells are added to the top-level design of the hierarchy during compile. New cells are added only to lower levels.

Default value for this variable is “false”.

**compile_preserve_subdesign_interfaces**
When true, disables customization of logic external to a subdesign during compile, and preserves the subdesign interface. When false (the default), compile customizes the logic external to a subdesign based on the subdesign's internal logic.

Default value for this variable is “false”.

**compile_retime_license_behavior**
Controls how the compile command behaves when the optimize_registers or balance_registers attributes are set on a design or parts of a design and the required license(s) (BOA-BRT or DC-Expert) are not available immediately.

Default value for this variable is “wait”.
**compile_sequential_area_recovery**

When true, compile remaps the sequential elements in the design to try and recover area. In doing so, compile will not worsen the delay cost (delay violation on the most critical path) of the design. However, the negative slack on violating, but noncritical paths, might be worsened.

Default value for this variable is “false”.

**compile_simple_mode_block_effort**

This variable is for use only with the set_simple_compile_mode command.

Default value for this variable is “none”.

**compile_top_all_paths**

This variable is for use only with the -top option of compile.

Default value for this variable is “false”.

**compile_update_annotated_delays_during_inplace_opt**

When true (the default value), reoptimize_design -in_place and compile -in_place are 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.

Default value for this variable is “true”.
**compile\_use\_fast\_delay\_mode**

When true, turns on delay calculation techniques that improve compile run times when the CMOS2 or nonlinear delay models are being used. The default is true. The improvements have the greatest impact when high fanout nets are encountered, as is often the case during sequential mapping.

Default value for this variable is “true”.

**compile\_use\_low\_timing\_effort**

Certain target libraries specify that the transition time (slew) at a gate's input pin may affect the output transition time of the gate. This means that local changes to a netlist may affect gate delays in the entire transitive fanout of the changed cells, requiring significant delay calculation and path tracing overhead. By default, all gate delays are calculated exactly during optimization at the expense of greater runtime. Timing effort is 'high'.

Default value for this variable is “false”.

**context\_check\_status**

One of a pair of status variables, syntax_check\_status and context_check\_status, whose values are set by the Syntax Checker and not by the user. You examine these variables to determine the status of the syntax_check or context_check mode of the Syntax Checker. A value of true indicates that the mode is currently enabled; a value of false indicates that the mode is disabled. For example, a value of context_check\_status = true indicates that the context_check mode is currently enabled. Both modes cannot be enabled simultaneously; these two status variables allow you to determine whether one mode is enabled before you attempt to enable the other mode.
**create_clock_no_input_delay**
Affects delay propagation characteristics of clock sources created using create_clock. When create_clock_no_input_delay is false (the default value), clock sources used in the data path are established as timing startpoints. The clock sources in the design will propagate rising delays on every rising clock edge, and will propagate falling delays on every falling clock edge. You can disable this behavior by setting create_clock_no_input_delay to true.

Default value for this variable is “false”.

**current_design**
Specifies the design being worked on. This variable is used by most of the Synopsys commands.

Default value for this variable is “”.

**db2sge_bit_type**
Specifies the name of the single bit type to be used by db2sge while transferring the type_conversion information to SGE.

Default value for this variable is “std_logic”.

**db2sge_bit_vector_type**
Specifies the name of the bit_vector type to be used by db2sge while transferring the type_conversion information to SGE.

Default value for this variable is “std_logic_vector”.

**db2sge_command**
Specifies the full path of the db2sge executable to use to transfer designs to SGE.

Default value for this variable is “synopsys_root”.

---

158 Synthesis Variables
**db2sge_display_instance_names**
Tells db2sge to display instance names in schematics.
Default value for this variable is “false”.

**db2sge_display_pin_names**
Tells db2sge to display pin names in schematics.
Default value for this variable is “false”.

**db2sge_display_symbol_names**
Tells db2sge to display symbol names in schematics.
Default value for this variable is “false”.

**db2sge_one_name**
Specifies the value of the logic one, in the enumerated type, to be used by db2sge while transferring the type_conversion information to SGE.
Default value for this variable is “1”.

**db2sge_output_directory**
Specifies the name of the directory(library) in which the symbols and schematics should be written. Db2sge adds the path to this directory(library) at the head of the list of Project Libraries in the .synopsys_sge.setup file in the current directory while transferring designs.
Default value for this variable is “”.

**db2sge_overwrite**
Db2sge overwrites existing symbol (.sym) or schematic (.sch) files if this is ‘true’.
Default value for this variable is “true”.

---

Synthesis Variables 159
**db2sge_scale**
Specifies the scale factor for symbols and schematics. The scale factor is used to prevent resolution problems when transferring symbols and schematics.
Default value for this variable is “2”.

**db2sge_script**
Specifies the name of the dc_shell script to use to transfer the design(s) to SGE. To use a different dc_shell script, change the value of this variable to point to the new dc_shell script.
Default value for this variable is “synopsys_root”.

**db2sge_target_xp**
When true, db2sge places an xp attribute on those symbols that have a corresponding schematic. This attribute is used by the VHDL netlister in SGE.
Default value for this variable is “false”.

**db2sge_tcf_package_file**
Specifies the name of the file to which db2sge will write the package of type conversion functions.
Default value for this variable is “synopsys_tcf.vhd”.

**db2sge_unknown_name**
Specifies the value of the unknown logic value, in the enumerated type, to be used by db2sge while transferring the type_conversion information to SGE.
Default value for this variable is “X”.

**db2sge_use_bustaps**
Replaces ripper symbols created in by create_schematic with bustaps in the SGE file when set to 'true'. If it is set to 'false', ripper symbols are preserved in the schematic transferred to SGE.
Default value for this variable is “false”.

160 Synthesis Variables
**db2sge_useCompoundNames**
Controls the compound-net bus naming style. For example, if a bus $Z[0:1]$ consists of nets A and B, with this variable set to 'true', the name of the bus in SGE is A,B. When it is set to 'false', the bus name $Z[0:1]$ is used in SGE.

Default value for this variable is “true”.

**db2sge_useLibSection**
Specifies the use_lib section to use from the .synopsys_sge2vhdl.setup file when generating VHDL for a schematic.

Default value for this variable is “”.

**db2sge_zero_name**
Specifies the value of the logic zero, in the enumerated type, to be used by db2sge while transferring the type_conversion information to SGE.

Default value for this variable is “0”.

**dc_shell_mode**
This variable is set to the mode of the application currently running. The value of this variable can be either of default or tcl. Default value means the application is running in dch default mode. tcl value indicates the application is running in Tcl mode. The variable is read only.

**default_input_delay**
Specifies the global default input delay value to be used for environment propagation. This variable is used by derive_constraints command.

Default value for this variable is “30.0”.
**default_name_rules**
Contains the name of a name_rule to be used as a default by change_names if a name_rule is not specified using the -rules name_rules option. change_names 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, refer to the define_name_rules manual page.
Default value for this variable is “”.

**default_output_delay**
Specifies the global default output delay value to be used for environment propagation. This variable is used by derive_constraints command.
Default value for this variable is “30.0”.

**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. The default value for default_port_connection_class is universal.
Default value for this variable is “universal”.

**default_schematic_options**
Specifies options to use when schematics are generated. When set to -size infinite (default), the schematic for a design is displayed on a single page. (Used by the Design Analyzer.)
Default value for this variable is “-size infinite”.
**design_library_file**
Gives the name of a file that contains design library mappings. The format of the file is the same as that used by the Synopsys simulator.

Default for this variable is ".synopsys_vss.setup".

**designer**
Name of the current user. This name is displayed on the schematics.

Default value for this variable is "".

**disable_library_transition_degradationFp**
When 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.

Default value for this variable is "false".

**dpcm_arc_sense_mapping**
When true (the default), Design Compiler maps half unate arcs to preset/clear arcs for sequential cells. When false, Design Compiler leaves the arcs from DPCM as they are.

Default value for this variable is "true".
**dpcm_debuglevel**

Determines the level of debugging for Design Compiler. When set to level 0 (the default) or higher, Design Compiler calls DPCM to set the debug level. At level 1 or higher, Design Compiler turns on interrupt handling, which determines whether the tool terminated abnormally within DPCM. For level 3 or higher, Design Compiler displays all EXTERNAL calls made by DPCM.

Default value for this variable is “0”.

**dpcm_functionscope**

Allowed values are global (the default) and instance. When set to global, the value is cached in DPCM, and DPCM does not request FunctionalMode for every delay/slew calculation call. When set to instance, for every delay or slew calculation call, DPCM makes a call to the external function CurrentFunctionalMode. The behavior usually depends on the way DPCM is implemented.

Default value for this variable is “global”.

**dpcm_level**

Allowed values are performance (the default) and accurate. When set to performance, Design Compiler requests DPCM to do calculations in performance mode. When set to accurate, Design Compiler requests DPCM to do calculations in accurate mode. Usually, fewer calls are made in performance mode, and DPCM caches more values.

Default value for this variable is “performance”.
**dpcm_libraries**

Specifies the libraries of the link_path that use the DPCM delay calculation. By default, delay calculation is performed with the Synopsys library delays. Set this variable only if a DPCM library is available for the specified library.

Default value for this variable is “{your_library.db}”.

**dpcm_rulepath**

A list of paths, similar to a search path, for locating the DPCM main rule. Design Compiler sets it to the DCMRULEPATH unix variable used by the DPCM delay calculation. Set this variable only if a DPCM library is available for the libraries used by the current design, and use the path specified by the DPCM provider.

Default value for this variable is “{.="/libraries”}”.

**dpcm_rulespath**

A list of paths, similar to a search path, for locating the DPCM subrules. Design Compiler sets it to the DCMRULESPATH unix variable used by the DPCM delay calculation. Set this variable only if a DPCM library is available for the libraries used by the current design, and use the path specified by the DPCM provider.

Default value for this variable is “{.="/libraries”}”.

**dpcm_slewlimit**

Determines behavior when input pin slew exceeds library limits. When false (the default), Design Compiler allows DPCM to handle these cases. When true, Design Compiler limits the slew to the maximum slew.

Default value for this variable is “false”.
**dpcm_tablepath**
A list of paths, similar to a search path, for locating the DPCM tables. Design Compiler sets it to the DCMTABLEPATH unix variable used by the DPCM delay calculation. Set this variable only if a DPCM library is available for the libraries used by the current design, and use the path specified by the DPCM provider.

Default value for this variable is “{.+}/libraries”.

**dpcm_temperaturescope**
Allowed values are global (the default) and instance. When set to global, the value is cached in DPCM, and DPCM does not request Temperature for every delay/slew calculation call. When set to instance, for every delay/slew calculation call, DPCM makes a call to the external function CurrentTemperature. The behavior usually depends on the way DPCM is implemented.

Default value for this variable is “global”.

**dpcm_version**
Specifies the version of the API used by the DPCM delay calculation. The supported DPCM API versions are IEEE-P1481 (the default) and IBM.

Default value for this variable is “IEEE-P1481”.

**dpcm_voltagescope**
Allowed values are global (the default) and instance. When set to global, the value is cached in DPCM, and DPCM does not request RailVoltage values for every delay/slew calculation call. When set to instance, for every delay or slew calculation call, DPCM makes a call to the external function RailVoltage. The behavior usually depends on the way DPCM is implemented.

Default value for this variable is “global”.

166 Synthesis Variables
**dpcm_wireloadslope**
Allowed values are global (the default) and instance. When set to global, the value is cached in DPCM, and DPCM does not request Current WireLoadModel values for every delay/slew calculation call. When set to instance, for every delay or slew calculation call, DPCM makes a call to the external function CurrentWireLoadModel. The behavior usually depends on the way DPCM is implemented.
Default value for this variable is “global”.

**duplicate_ports**
Partitioning option that specifies if ports are to be drawn on every sheet for which an input or output signal appears. When true, no off-sheet connectors are used for input and output signals, and signal ports are duplicated where indicated on each sheet.
Default value for this variable is “false”.

**echo_include_commands**
When true, the include command prints the contents of files it executes. Otherwise, contents are not printed.
Default value for this variable is “true”.

**eco_align_design_verbose**
When true, causes more explicit messages to be displayed by the eco_align_design command. The default is false.

Default value for this variable is “false”.

---

Synthesis Variables 167
eco_allow_register_type_difference
   When true, causes eco_implement to allow type
differences between aligned sequential cells after an
attempt to remap. When false (the default), if the
difference cannot be resolved by remapping,
eco_implement terminates with an error message.
   Default value for this variable is “false”.

eco_correspondence_analysis_verbose
   When true, causes more explicit messages to be
displayed during the correspondence analysis
(structural and functional) during eco_implement.
The default is false.
   Default value for this variable is “false”.

eco_directives_verbose
   When true, causes more explicit messages to be
displayed during execution of the commands that
specify the directives to ECO Compiler, listed under
SEE ALSO. The default is false.
   Default value for this variable is “false”.

eco_implement_effort_level
   Determines the effort level to be used by
eco_implement in determining
functionally-equivalent end points between old and
new netlists. Allowed values are low (the default) and
high.
   Default value for this variable is “low”.

eco_instance_name_prefix
   Specifies the prefix of the names used to name the
cells and nets of unreused logic in the new
implementation (eco design) after eco_implement.
The default is eco_
   Default value for this variable is “eco_”.

168 Synthesis Variables
**eco_remap_register_verbose**

When true, causes more explicit messages to be displayed during the register mapping phase of eco_implement. The default is false.

Default value for this variable is “false”.

**eco_reuse_verbose**

When true, causes more explicit messages to be displayed during the identification of reusable components phase for eco_implement. The default is false.

Default value for this variable is “false”.

**edifin_autoconnect_offpageconnectors**

When true, connects an off-page connector to the port that has the same name (if there is one), even if there is no explicit connection statement. Usually, this connection should not be made without an explicit connection statement, nor should there be a port and an off-page connector with the same name. When false (the default value), a file containing an off-page connector and a port with the same name cannot be read at all. Some EDIF writers do not include explicit connections between off-page connectors and ports, but do give them the same names.

Default value for this variable is “false”.

**edifin_autoconnect_ports**

When true, connects a port to a net with the same name (if there is one), even if there is no explicit connection statement. Usually, this connection should not be made without an explicit connection statement. Some EDIF writers do not include explicit connections between ports and nets, but do give them the same names.

Default value for this variable is “false”.

---

Synthesis Variables 169
**edifin_dc_script_flag**
When set to a non-empty string, sends commands, embedded as comments in the EDIF file, to Design Compiler. To indicate that a comment contains commands, the first argument in the comment construct must be a string with the value of this variable. The second argument must be a string comprising the command. When this variable is not set, EDIF comments are ignored during read.
Default value for this variable is “”.

**edifin_delete_empty_cells**
When true, deletes all designs just read that do not contain cells or nets.
Default value for this variable is “true”.

**edifin_delete_ripper_cells**
When true, deletes all designs just read of cellType RIPPER.
Default value for this variable is “true”.

**edifin_ground_net_name**
Specifies EDIF ground nets. If you set this variable to the name of an EDIF ground net, the net is recognized by Synopsys software as a ground net.
Default value for this variable is “”.

**edifin_ground_net_property_name**
Specifies EDIF ground nets. If you set this variable and edifin_ground_net_property_value equal to the property name and property value of an EDIF ground net, the net is recognized by Synopsys software as a ground net.
Default value for this variable is “”.
edifin_ground_net_property_value
Specifies EDIF ground nets. If you set edifin_ground_net_property_name and this variable equal to the property name and property value of an EDIF ground net, the net is recognized by Synopsys software as a ground net.
Default value for this variable is "".

edifin_ground_port_name
Specifies EDIF ground ports. If you set this variable to the name of an EDIF ground port, the port is recognized by Synopsys software as a ground port.
Default value for this variable is "".

edifin_instance_property_name
If a property on an instance construct has this name, and if the property value is a string, that value is assigned to the cell_property attribute of the cell instance.
Default value for this variable is "".

edifin_lib_in_osc_symbol
Affects EDIF read_lib . If a cell with this name is found in the EDIF symbol library, it is declared as the name of the input off-sheet connector symbol in the Synopsys intermediate symbol library source file and Synopsys symbol library file.
Default value for this variable is "".

edifin_lib_in_port_symbol
Affects EDIF read_lib . If a cell with this name is found in the EDIF symbol library, it is declared as the name of the input port symbol in the Synopsys intermediate symbol library source file and Synopsys symbol library file.
Default value for this variable is "".
**edifin_lib_inout_osc_symbol**
Affects EDIF read_lib. If a cell with this name is found in the EDIF symbol library, it is declared as the name of the input/output off-sheet connector symbol in the Synopsys intermediate symbol library source file and Synopsys symbol library file.
Default value for this variable is "".

**edifin_lib_inout_port_symbol**
Affects EDIF read_lib. If a cell with this name is found in the EDIF symbol library, it is declared as the name of the input/output port symbol in the Synopsys intermediate symbol library source file and Synopsys symbol library file.
Default value for this variable is "".

**edifin_lib_logic_0_symbol**
Affects EDIF read_lib. If a cell with this name is found in the EDIF symbol library, it is declared as the name of the ground symbol in the Synopsys intermediate symbol library source file and Synopsys symbol library file.
Default value for this variable is "".

**edifin_lib_logic_1_symbol**
Affects EDIF read_lib. If a cell with this name is found in the EDIF symbol library, it is declared as the name of the power symbol in the Synopsys intermediate symbol library source file and Synopsys symbol library file.
Default value for this variable is "".
edifin_lib_mentor_netcon_symbol
Affects EDIF read_lib. If a cell with this name is found in the EDIF symbol library, it is declared as the name of the Mentor $netcon symbol in the Synopsys intermediate symbol library source file and Synopsys symbol library file.

Default value for this variable is "".

edifin_lib_out_osc_symbol
Affects EDIF read_lib. If a cell with this name is found in the EDIF symbol library, it is declared as the name of the output off-sheet connector symbol in the Synopsys intermediate symbol library source file and Synopsys symbol library file.

Default value for this variable is "".

edifin_lib_out_port_symbol
Affects EDIF read_lib. If a cell with this name is found in the EDIF symbol library, it is declared as the name of the output port symbol in the Synopsys intermediate symbol library source file and Synopsys symbol library file.

Default value for this variable is "".
**edifin_lib_ripper_bits_property**
Affects EDIF read_lib. If a cell with the name specified by the variable `edifin_lib_ripper_cell_name` is found in the EDIF symbol library, the value of this variable is assigned to the `ripped_bits_property` attribute of this ripper symbol in the Synopsys intermediate symbol library source file and Synopsys symbol library file. Or if no cell name is specified by the variable `edifin_lib_ripper_cell_name`, if a cell of cellType RIPPER is found in the EDIF symbol library, the value of this variable is assigned to the attribute of this ripper symbol. In EDIF files written with this ripper symbol, the value of this attribute is the name of a property that is placed on each ripper instance; the number of the bit (or range of bits) is the property's value.

Default value for this variable is "".

**edifin_lib_ripper_bus_end**
Affects EDIF read_lib. If a cell with the name specified by the variable `edifin_lib_ripper_cell_name` is found in the EDIF symbol library, the value of this variable is assigned to the `ripped_pin` attribute of this ripper symbol in the Synopsys intermediate symbol library source file and Synopsys symbol library file. Or, if no cell name is specified by the variable `edifin_lib_ripper_cell_name`, if a cell of cellType RIPPER is found in the EDIF symbol library, the value of this variable is assigned to the `ripped_pin` attribute of this ripper symbol. In EDIF files written with instances of this ripper symbol, this attribute specifies the bus end pin of the ripper. If no pin exists on the ripper symbol with the name specified by this variable, results might be erroneous.

Default value for this variable is "".
**edifin_lib_ripper_cell_name**
Affects EDIF read_lib. If a cell with this name is found in the EDIF symbol library, the values of the variables `edifin_lib_ripper_bits_property` and `edifin_lib_ripper_bus_end` are assigned to the ripped_bits_property and ripped_pin attributes of this ripper symbol in the Synopsys intermediate symbol library source file and Synopsys symbol library file. Or, if no cell name is specified by this variable, if a cell of cellType RIPPER is found in the EDIF symbol library, the values of the variables `edifin_lib_ripper_bits_property` and `edifin_lib_ripper_bus_end` are assigned to the ripped_bits_property and ripped_pin attributes of this ripper symbol.

Default value for this variable is "".

**edifin_lib_ripper_view_name**
Affects EDIF read_lib. This variable affects the reading of a cell with the name specified by the variable `edifin_lib_ripper_cell_name`. Or, if no cell name is specified by the variable `edifin_lib_ripper_cell_name`, this variable affects the reading of a cell of cellType RIPPER. Only the view with this name is added as this ripper symbol in the Synopsys intermediate symbol library source file and Synopsys symbol library file. If the value of this variable is not specified, only the first view of the cell is added as this ripper symbol.

Default value for this variable is "".
**edifin_lib_route_grid**
Affects EDIF read_lib. The value of this variable is declared as the size of the route grid of the library in the Synopsys intermediate symbol library source file and Synopsys symbol library file.

Default value for this variable is “1024”.

**edifin_lib_templates**
Affects EDIF read_lib. This variable is a list of sublists. Each sublist contains three elements: first, a string that defines a sheet size name; second, a string that defines the template orientation (either landscape or portrait); third, a string that defines the symbol name corresponding to the sheet size specified by the first element.

Default value for this variable is “{ }”.

**edifin_portinstance_disabled_property_name**
If the property of a portInstance construct has the same name as this variable string, and if the property's value is the same as edifin_portinstance_disabled_property_value, the disabled attribute is placed on the pin of the cell instance.

Default value for this variable is “”.

**edifin_portinstance_disabled_property_value**
If the property of a portInstance construct has the same name as edifin_portinstance_disabled_property_name, and if the property's value is the same as this variable string, the disabled attribute is placed on the pin of the cell instance.
**edifin_portinstance_property_name**
If the property of a portInstance construct has the same name as this variable string, and if the property's value is a string, that value is assigned to the pin_properties attribute of the pin of the cell instance.

Default value for this variable is "".

**edifin_power_net_name**
Specifies EDIF power nets. If you set this variable to the name of an EDIF power net, the net is recognized by Synopsys software as a power net.

Default value for this variable is "".

**edifin_power_net_property_name**
If the values of this variable and edifin_power_net_property_value are the same as the property name and property value of an EDIF net, the net is recognized as a power net.

Default value for this variable is "".

**edifin_power_net_property_value**
If the values of edifin_power_net_property_name and this variable are the same as the property name and property value of an EDIF net, the net is recognized as a power net.

Default value for this variable is "".

**edifin_power_port_name**
Specifies EDIF power ports. If you set this variable to the name of an EDIF power port, the port is recognized by Synopsys software as a power port.

Default value for this variable is "".
**edifin_use_identifier_in_rename**
When true, gives all objects created the name specified by the identifier in the name construct (which might be made up of a rename construct). When false, gives all objects created the name specified by the name in the rename construct (if the name construct is made up of a rename construct). Also affects EDIF read_lib.

Default value for this variable is “false”.

**edifin_view_identifier_property_name**
Allows multiple views (representations) of a cell to be created in the Synopsys database. This variable specifies an EDIF property whose value becomes the name of the cell object created. When set to an empty string, only the first cell view is recognized and subsequent views are ignored. (Also affects EDIF read_lib.)

Default value for this variable is “”.

**edifout_dc_script_flag**
When set to a non-empty string, embeds comments containing Design Compiler commands into the EDIF file.

Default value for this variable is “”.

**edifout_design_name**
Controls the identifier in the design construct written at the end of EDIF files. When you set this variable, any non-alphanumeric character in the string is changed to an underscore. This construct appears only at the end of an EDIF file written out.

Default value for this variable is “Synopsys_edif”.

---

178 Synthesis Variables
**edifout_designs_library_name**

Specifies the name that is given to the EDIF library construct that contains the cell constructs for the designs being written.

Default value for this variable is “DESIGNS”.

**edifout_display_instance_names**

When true, displays cell instance names in the schematic.

Default value for this variable is “false”.

**edifout_display_net_names**

When true, displays the net name at each port, pin, and off-sheet connector location in the schematic. Two display constructs are included in the EDIF file for each port, pin, and off-sheet connector on a net.

Default value for this variable is “false”.

**edifout_external**

Usually, complete definitions are written for all symbols used in an EDIF schematic. When true, only the interface for each referenced symbol is written in an external construct. (Symbol definitions are not included.)

Default value for this variable is “true”.

**edifout_external_graphic_view_name**

Specifies the name of an EDIF view used in a library contained in an external construct in a schematic EDIF file. This variable affects only symbols with no pins such as port or off-sheet connector symbols. Set this variable if your target EDIF reader expects specific view names for cells of type GRAPHIC.

Default value for this variable is “Graphic_representation”.
**edifout_external_netlist_view_name**
Specifies the name of an EDIF view used in a library contained in an external construct in a netlist EDIF file. Set this variable if your target EDIF reader expects specific view names for cells of type NETLIST.

**edifout_external_schematic_view_name**
Specifies the name of an EDIF view used in a library contained in an external construct in a schematic EDIF file. This variable affects schematics and symbols, except for symbols with no pins such as port or off-sheet connector symbols. Set this variable if your target EDIF reader expects specific view names for cells of type SCHEMATIC.

Default value for this variable is “Schematic_representation”.

**edifout_ground_name**
When edifout_power_and_ground_representation is set to cell and edifout_netlist_only is true, this is the name given to the built-in Synopsys ground cell. This variable is used in conjunction with edifout_ground_pin_name, which names the ground cell’s pin.

Default value for this variable is “logic_0”.

**edifout_ground_net_name**
When edifout_power_and_ground_representation is set to net, and if this variable and the variable edifout_power_net_name are non-empty strings, the ground nets are written as one net. This variable specifies the name of the ground net.

Default value for this variable is “”.
edifout_ground_net_property_name
When edifout_power_and_ground_representation is set to net, and if this variable and the variables edifout_ground_net_property_value, edifout_power_net_property_name, and edifout_power_net_property_value are non-empty strings, this variable specifies the name of the property used to identify the ground nets (or the one ground net created by the variable edifout_ground_net_name).
Default value for this variable is "".

edifout_ground_net_property_value
When edifout_power_and_ground_representation is set to net, and if this variable and the variables edifout_ground_net_property_name, edifout_power_net_property_name, and edifout_power_net_property_value are non-empty strings, this variable specifies the value of the property used to identify the ground nets (or the one ground net created by the variable edifout_ground_net_name).
Default value for this variable is "".

edifout_ground_pin_name
When edifout_power_and_ground_representation is set to cell and edifout_netlist_only is true, this is the name given to the pin of the ground cell.
Default value for this variable is “logic_0_pin”.
**edifout_ground_port_name**

When `edifout_power_and_ground_representation` is set to `portf P`, two extra port constructs are included in the interface of every cell: the power port and the ground port. This is the name given to the ground port.

Default value for this variable is “GND”.

**edifout_instance_property_name**

If a cell instance has the `cell_property` attribute, a property with the same name as this variable and a value the same as the `cell_property` attribute, is included on the EDIF instance.

Default value for this variable is “”. 

**edifout_instantiate_ports**

When true, symbol constructs for ports and off-sheet connectors are included in a library (or external) construct in an EDIF schematic. In addition, instantiations referencing those symbols are included in the portImplementation constructs for ports and off-sheet connectors in the contents constructs of EDIF schematics. When false, schematic descriptions are included in the portImplementation constructs for ports and off-sheet connectors in the contents constructs of EDIF schematics. (This variable does not affect the way portImplementation constructs are written in symbol constructs.)

Default value for this variable is “false”.
**edifout_library_graphic_view_name**

Specifies the name of an EDIF view used in a library contained in a library construct (not an external construct) in a schematic EDIF file. This variable affects only symbols with no pins, such as port or off-sheet connector symbols. Set this variable if your target EDIF reader expects specific view names for cells of type GRAPHIC.

Default value for this variable is “Graphic_representation”.

**edifout_library_netlist_view_name**

Specifies the name of an EDIF view used in a library contained in a library construct (not an external construct) in a netlist EDIF file. Set this variable if your target EDIF reader expects specific view names for cells of type NETLIST.

Default value for this variable is “Netlist_representation”.

**edifout_library_schematic_view_name**

Specifies the name of an EDIF view used in a library contained in a library construct (not an external construct) in a schematic EDIF file. This variable affects schematics and symbols, except for symbols with no pins, such as port or off-sheet connector symbols. Set this variable if your target EDIF reader expects specific view names for cells of type SCHEMATIC.

Default value for this variable is “Schematic_representation”.
**edifout_merge_libraries**

When true, writes all cells for an EDIF file in the same library. This causes problems if there is more than one cell with the same name. This variable is included for EDIF readers that can handle only a single library per file.

Default value for this variable is “false”.

**edifout_multidimension_arrays**

When this variable is true, if the structure of a bus makes it possible to represent that bus as a multi-dimensional array, it will be represented as a multi-dimensional array.

Default value for this variable is “false”.

**edifout_name_oscs_different_from_ports**

When true, each off-sheet connector that is connected to a port is given a different name from the port in the descriptions of designs written in EDIF format. (Other off-sheet connectors might be renamed to avoid creating shorts.) Off-sheet connector names in Design Compiler are unchanged.

**edifout_name_rippers_same_as_wires**

When false, bus ripper cell instances are named in the order they are created during schematic generation. The name for the ripper instance is Ripper_n, where n is the incrementing integer value for each bus ripper cell instance created. For example, the first ripper instance in the schematic is named:

Ripper_1

The second instance is named:

Ripper_2

and so on.

Default value for this variable is “false”.
**edifout_netlist_only**
When true, writes an EDIF netlist, not an EDIF schematic.

Default value for this variable is “false”.

**edifout_no_array**
When this variable and edifout_netlist_only are true, no array constructs are written. In designs containing busses, individual bus members are written instead.

Default value for this variable is “false”.

**edifout_numerical_array_members**
When edifout_netlist_only is false or this variable is false, in member constructs, the integerValue that is the index into an array is 0 for the first member of the array and increases by 1 for each succeeding member of the array, regardless of whether the bus is ascending or descending.

Default value for this variable is “false”.

**edifout_pin_direction_in_value**
Includes an EDIF property with the same name as edifout_pin_direction_property_name, and a value the same as this variable string, on input pins.

Default value for this variable is “"".

**edifout_pin_direction_inout_value**
Includes an EDIF property with the same name as edifout_pin_direction_property_name, and a value the same as this variable, on input/output pins.

Default value for this variable is “"".
**edifout_pin_direction_out_value**
Includes an EDIF property with the same name as edifout_pin_direction_property_name, and the value the same as this variable, on output pins.

Default value for this variable is "".

**edifout_pin_direction_property_name**
Includes an EDIF property on all pins. The name of the property is the same as this variable string. To determine the value of the property, this variable is used in conjunction with edifout_pin_direction_in_value, edifout_pin_direction_inout_value, and edifout_pin_direction_out_value.

Default value for this variable is "".

**edifout_pin_name_property_name**
Includes on pins an EDIF property with the same name as this variable string, and value the same as the pin name.

Default value for this variable is "".

**edifout_portinstance_disabled_property_name**
If the pin of a cell instance has the disabled attribute, a property with the same name as this variable string and a value the same as edifout_portinstance_disabled_property_value is created in the portInstance construct.

Default value for this variable is "".
edifout_portinstance_disabled_property_value
If the pin of a cell instance has the disabled attribute, a property with the same name as
edifout_portinstance_disabled_property_name and value the same as this variable string is created in the portInstance construct.
Default value for this variable is “”.

edifout_portinstance_property_name
If the pin_properties attribute is on the pin of a cell instance, a property with the same name as this variable string and with a value the same as the pin_properties attribute is created in the portInstance construct.
Default value for this variable is “”.

edifout_power_and_ground_representation
Determines power and ground representations in EDIF files. This variable can have the value port, cell (the default), or net.
Default value for this variable is “cell”.

edifout_power_name
When edifout_power_and_ground_representation is set to cell, and edifout_netlist_only is true, this is the name given to the built-in Synopsys power cell. This variable is used in conjunction with edifout_power_pin_name, which names the power cell's pin.
Default value for this variable is “logic_1”.
**edifout_power_net_name**

When `edifout_power_and_ground_representation` is set to net, and if this variable and the variable `edifout_ground_net_name` are non-empty strings, the power nets are written as one net. This variable specifies the name of the power net.

Default value for this variable is “”.

**edifout_power_net_property_name**

When `edifout_power_and_ground_representation` is set to net, and if this variable and the variables `edifout_ground_net_property_name`, `edifout_ground_net_property_value`, and `edifout_power_net_property_value` are non-empty strings, this variable specifies the name of the property used to identify the power nets (or the one power net created by the variable `edifout_power_net_name`).

Default value for this variable is “”.

**edifout_power_net_property_value**

When `edifout_power_and_ground_representation` is set to net, and if this variable and the variables `edifout_ground_net_property_name`, `edifout_ground_net_property_value`, and `edifout_power_net_property_name` are non-empty strings, this variable specifies the value of the property used to identify the power nets (or the one power net created by the variable `edifout_power_net_name`).

Default value for this variable is “”.
**edifout_power_pin_name**
When edifout_power_and_ground_representation is set to cell, and edifout_netlist_only is true, this is the name given to the pin of the power cell.

Default value for this variable is “logic_1_pin”.

**edifout_power_port_name**
When edifout_power_and_ground_representation is set to port, two extra port constructs are included in the interface of every cell: the power port and the ground port. This is the name given to the power port.

Default value for this variable is “VDD”.

**edifout_skip_port_implementations**
When true, does not write portImplementation constructs for ports in the contents constructs of EDIF schematics. Note that portImplementation constructs are still written in symbol constructs, regardless of the value of this variable.

Default value for this variable is “false”.

**edifout_target_system**
Used so that the values of related edifout variables are overridden so that object names can be transferred to the target system, and to perform vendor-specific operations beyond the scope of edifout variables. The variable can be set to mentor, valid, or cadence. If you are using mentor or valid, consult the appropriate EDIF Interface Application Note for more details.

Default value for this variable is “”.

**edifout_top_level_symbol**
When true, writes the top-level symbol in EDIF schematic files. To disable the top-level symbol, set this variable to false.

Default value for this variable is “true”.
**edifout_translate_origin**

When set to center or lower-left, the values of the coordinates are transformed in schematic descriptions to reflect the specified origin. When this variable is set to center, the center of each sheet is at the origin. When this variable is set to lower-left, the lower left corner of each sheet is at the origin, therefore all x- and y-coordinates are non-negative.

Default value for this variable is “"”.

**edifout_unused_property_value**

Used when writing out EDIF netlists to be processed by a Motorola EDIF reader. EDIF netlists are required to have the unused attribute attached to unconnected port instances (pins).

Default value for this variable is “"”.

**edifout_write_attributes**

When true, embeds comments containing Design Compiler attribute definitions into the EDIF file. All active attribute definitions in the design during the current Design Compiler session are written out. This variable is used in conjunction with edifout_dc_script_flag.

Default value for this variable is “false”.

**edifout_write_constraints**

When true, embeds comments containing Design Compiler constraint commands into the EDIF file. All active constraints on the design during the current Design Compiler session are written out. This variable is used in conjunction with edifout_dc_script_flag.

Default value for this variable is “false”.

---

190 Synthesis Variables
**edifout_write_properties_list**
Specifies a list of library, cell, or port properties to write into the EDIF description. The value of this variable is a list of strings containing the property names.

Default value for this variable is “{ }”.

**enable_instances_in_report_net**
To enable report_net to report on instances in the current design, set enable_instances_in_report_net to true.

Default value for this variable is “false”.

**enable_page_mode**
When true, long reports are displayed one page at a time (similar to the UNIX more command). Commands affected by this variable include list, help, and the report commands.

Default value for this variable is “true”.

**enable_recovery_removal_arcs**
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 has to be held after the closing edge of clock.

Default value for this variable is “false”.

**equationout_and_sign**
Specifies the “and: sign to use when writing a design in equation format. It must be an * (asterisk) or & (ampersand). If you specify an invalid value, the default is used.

Default value for this variable is “*”.

---

*Synthesis Variables* 191
**equationout_or_sign**

Specifies the “or” sign to use when writing a design in equation format. It must be a + (plus) or | (vertical bar). If you specify an invalid value, the default is used.

Default value for this variable is “+”.

**equationout_postfix_negation**

When true, the ‘ (single quote) is used as the negation operator when writing a design in equation format. When false, the prefix negation operator ! (exclamation mark) is used.

Default value for this variable is “true”.

**exit_delete_filename_log_file**

When true (the default value), causes the file specified by the variable filename_log_file to be deleted after design_analyzer or dc_shell exits normally. Set exit_delete_filename_log_file to false if you want the file to be retained.

Default value for this variable is “true”.

**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. The file specified by filename_log_file will contain all the filenames read in by design_analyzer or dc_shell, including db, script, verilog, vhdl or 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 files.log will be used. This file will be deleted if the program exits normally, unless exit_delete_filename_log_file is set to false.

Default value for this variable is “filenames.log”.
find_converts_name_lists
When true, directs the find command to convert 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 and would convert the string instance to a list of strings before searching for objects.

Default value for this variable is “false”.

gen_bussing_exact_implicit
When true, specifies that schematics generated using create_schematic -implicit should contain no bus rippers. Instead, all bussed connections should be shown with implicit bus names. This should be used with the -implicit command line option. By default, schematics generated using the -implicit option have rippers between any bus connections where the riper connects to a pin in a column adjacent to the originating bus pin. Bussed connections between cell pins more than one column away from each other are always shown disconnected with the -implicit option.

Default value for this variable is “false”.

gen_cell_pin_name_separator
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, / is used to separate the cell and pin names, thus creating bus names like U0/OUT[0:3]

Default value for this variable is “/”.

---

Synthesis Variables 193
**gen_create_netlist_busses**

When true, create_schematic will create netlist busses whenever it creates busses on the schematic. Usually, this happens when create_schematic creates busses to connect to bussed pins on the schematic, but it may also happen when there are bussed ports on the schematic. The default is false.

Default value for this variable is “false”.

**gen_dont_show_single_bit_busses**

This variable is used in conjunction with the gen_show_created_busses variable. When gen_show_created_busses is true and gen_dont_show_single_bit_busses is also set to true, single bit busses created by gen are not printed out. This suppresses messages about creation of single bit busses in the schematic.

Default value for this variable is “false”.

**gen_match_ripper_wire_widths**

When true, specifies that create_schematic generates rippers such that the width of the ripper always equals the width of the ripped net. When this variable is set to true, any rippers whose wire ends are connected to scalar nets will be of unit width. The default is false.

Default value for this variable is “false”.

---

194 Synthesis Variables
**gen_max_compound_name_length**

For the -sge create_schematic option, this variable controls the maximum length for compound names of bus bundles. The default for this is 256, which is the maximum length supported by SGE. Any busses with names longer than this variable are decomposed into their individual members in the schematic. If any such busses connect to cells referencing library symbols, those library symbols are ignored, and instead, new gen-created symbols are used. These gen created symbols have any busses whose compound names exceed this length blown up into their individual members.

Default value for this variable is “256”.

**gen_max_ports_on_symbol_side**

Specifies the maximum allowed size of a symbol created by create_schematic. For example, if this variable is five, symbols with no more than five ports on any one side are created. If this variable is not set or is set to zero, all input ports are placed on the left side of the symbol, and all inout and output ports are on the right.

Default value for this variable is “0”.

**gen_open_name_postfix**

Specifies the postfix to be used by create_schematic -sge when creating placeholder net names for unconnected pins.

Default value for this variable is “”.

**gen_open_name_prefix**

Specifies the prefix to be used by create_schematic -sge when creating placeholder net names for unconnected pins. The default is Open.

Default value for this variable is “Open”.

---

Synthesis Variables 195
**gen_show_created_busses**
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.

Default value for this variable is “false”.

**gen_show_created_symbols**
When this variable is set to true, 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.

Default value for this variable is “false”.

**gen_single_osc_per_name**
When this variable is set to true, only a 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 have their net segments show up as unconnected stubs.

Default value for this variable is “false”.

**generic_symbol_library**
The db file that contains generic symbols, templates, and layers used for schematics.

Default value for this variable is “generic.sdb”.
**hdl_keep_licenses**
When true (the default value), hdl licenses that are checked out remain checked out throughout the dc_shell session. When this variable is false, hdl licenses are released during execution of compile, after compile has completed the subtasks that require the license.

Default value for this variable is “true”.

**hdl_naming_threshold**
Determines the maximum character length that a parameter can have in order to be included in the design name; the default is 20. Normally, design names include the values of the design's parameters/generics. Parameters with a character length greater than the current value of this variable are omitted from the design name. Currently, this variable takes effect only when there is at least one non-integer parameter.

Default value for this variable is “20”.

**hdl_preferred_license**
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.

Default value for this variable is “”.

**hdlin_auto_save_templates**
When true, HDL designs containing parameters are read in as templates. HDL parameter files include Verilog modules with parameter declarations, and VHDL entities with generic declarations.

Default value for this variable is “false”.
**hdlin_check_no_latch**
When true, a warning message is issued if a latch is inferred from a design. This is useful for verifying that a combinational design does not contain memory components. The default is false.

Default value for this variable is “FALSE”.

**hdlin_dont_infer_mux_for_resource_sharing**
When true (the default value), HDL Compiler does not infer a MUX_OP for a signal/variable assigned in a case statement if the signal/variable is driven by a synthetic operator in more than one branch. When false, a MUX_OP may be inferred, and the synthetic operators driving the signal are not sharable.

Default value for this variable is “true”.

**hdlin_enable_analysis_info**
dc_shell variable that enables the generation of RTL_Analyzer analysis information.

When set to true, RTL Analyzer analysis information is created for designs processed by subsequent dc_shell commands. In the Synopsys RTL Analyzer tool, analysis 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 following dc_shell commands: analyze, elaborate, and rtl_analyzer.

Default value for this variable is “true”.
**hdlin_enable_rtdrc_info**

dc_shell variable that enables the generation of file name and line number information for RTL TestDRC.

When set to true, file name and line number information for HDL constructs and instances is created for designs processed by subsequent dc_shell commands. In the Synopsys RTL 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 dc_shell command rtlargc.

Default value for this variable is “true”.

**hdlin_enable_vpp**

The hdlin_enable_vpp variable controls Verilog Preprocessing (VPP). VPP support includes `ifdef, `else, `endif, and enhanced support of `define. With VPP support enabled, `define can be used in all contexts (example: `WIDTH'b0 is supported). The enhanced `define support also supports passing arguments to macros. The features of the Verilog Preprocessor are enabled when this variable is set to true. Setting hdlin_enable_vpp to false (the default) disables VPP.

Default value for this variable is “false”.

---

Synthesis Variables 199
**hdlin_ff_always_async_set_reset**

When true (the default value), the HDL Compiler checks and reports 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; you do not want asynchronous set or reset devices inferred; and/or you want your design to be input faster.

Default value for this variable is “true”.

**hdlin_ff_always_sync_set_reset**

When true, for a design subsequently analyzed, 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. The default is false.

Default value for this variable is “FALSE”.

**hdlin_hide_resource_line_numbers**

When true, prevents (V)HDL Compiler from appending the HDL source line number to the inferred cell's name when inferring a synthetic library or DesignWare part. When false (the default), the HDL source line number is automatically appended to the name of the inferred cell. Setting this variable to true makes the results of the Design Compiler compile command independent of the location of the inferred synthetic library or DesignWare parts in the HDL source.

Default value for this variable is “FALSE”.

---

200 Synthesis Variables
**hdlin_infer_multibit**

Specifies inference of multi-bit components for an entire design. The allowed values for `hdlin_infer_multibit` are `default_all`, `default_none`, and `never`.

Default value for this variable is “default_none”.

**hdlin_infer_mux**

Determines whether and how HDL Compiler infers a MUX_OP. Allowed values for `hdlin_infer_mux` are `all`, `none`, and `default`. When all, HDL Compiler attempts to infer a MUX_OP for a signal/variable assigned in a case statement. When none, HDL Compiler does not attempt to infer any MUX_OPs for a Verilog/VHDL design. When default (the default value), HDL Compiler attempts to infer a MUX_OP for a signal/variable assigned in a case statement if the statement is in a process associated with the `infer_mux` attribute/directive. A MUX_OP cannot be inferred if it violates the `hdlin_mux_size_limit` variable.

Default value for this variable is “default”.

**hdlin_keep_feedback**

When false (the default value), the HCL Compiler removes all flip-flop feedback loops. For example, feedback loops inferred from the statement `Q = Q` are removed. A simple D flip-flop with the state feedback removed becomes a synchronous load flip-flop. Set this variable to true if you want to keep feedback loops.

Default value for this variable is “FALSE”.
hdlin_keep_inv_feedback
When true (the default value), HDL Compiler keeps all inverted flip-flop feedback loops. When the variable is set to false, HDL Compiler removes all inverted flip-flop feedback loops; for example, feedback loops inferred from the statement \( Q = QN \) are removed and synthesized as T flip-flops. Set this variable to false if you want to infer Toggle or JK flip-flops.

Default value for this variable is “TRUE”.

hdlin_latch_always_async_set_reset
When true, for a design subsequently analyzed, every constant 0 loaded on a latch is used for asynchronous reset, and every constant 1 loaded on a latch, is used for asynchronous set. The default is false.

Default value for this variable is “FALSE”.

hdlin_merge_nested_conditional_statements
When false (the default value), HDL Compiler infers a SELECT_OP (a generic logic component inferred for conditional statements) for each if/or case statement in a nested conditional construct. This gives you the ability to structure the HDL for late arriving signals.

Default value for this variable is “false”.

hdlin_mux_oversize_ratio
An integer that prevents inference of a sparse multiplexor. When the ratio of MUX_OP data inputs to unique data inputs is above the hdlin_mux_oversize_ratio, a MUX_OP is not inferred. The default ratio is 100.

Default value for this variable is “100”.
**hdlin_mux_size_limit**
An integer that limits the number of inputs of an inferred multiplexer; the default is 32. HDL Compiler does not generate MUX_OPs for multiplexers specified with more than this maximum number of inputs. The size of a multiplexer can be impounded by nested if/or case statements.

Default value for this variable is “32”.

**hdlin_preserve_vpp_files**
Verilog Preprocessing creates files with a pp extension, which are removed after parsing. These files hold the result of evaluating `ifdef, `define, and the other preprocessor directives. The hdlin_preserve_vpp_files variable controls deletion of these files. When true, the intermediate preprocessor files remain after parsing. When false (the default), the intermediate VPP files are deleted once they are no longer needed.

Default value for this variable is “false”.

**hdlin_reg_report_length**
An integer variable that sets the maximum length, in characters, of the Boolean formulas reported when hdlin_report_inferred_modules is set to verbose. The default value is 60.

Default value for this variable is “60”.
**hdlin_replace_synthetic**
When true, synthetic library parts in HDL designs are processed during the read and elaborate commands; actual gate-level implementations are inserted during the read command. The processing is exactly the same as typing replace_synthetic -ungroup. A simple area-based resource sharing step is called on the design, followed by implementing all the parts with minimum area implementation.

Default value for this variable is “false”.

**hdlin_report_inferred_modules**
Possible values for hdlin_report_inferred_modules are -true, verbose, and false. When true (the default value), read or elaborate generates a brief report for inferred latches, flip-flops, three-state, and multiplexor devices. When verbose, detailed information also is reported, including the asynchronous set or reset, synchronous set or reset, and synchronous toggle conditions of each latch or flip-flop expressed in Boolean formulas. When false, no report is generated.

Default value for this variable is “true”.

**hdlin_translate_off_skip_text**
When true, the VHDL Compiler treats the text between the translation directives translate_off and translate_on as comments. The default is false. The directives synthesis_off and synthesis_on are not affected.

Default value for this variable is “false”.

---

204 Synthesis Variables
**hdlin_vpp_temporary_directory**

Verilog preprocessing constructs intermediate files, which are read during Verilog parsing then deleted. The `hdlin_vpp_temporary_directory` variable determines where these files are created. By default, this variable is set to an empty string, and files are created in the WORK directory. Setting this variable to a directory path causes intermediate files to be created in the specified directory. Setting `hdlin_vpp_temporary_directory` restores the default behavior of using the WORK directory.

Default value for this variable is “/tmp/vpp”.

**hdlin_write_gtech_design_directory**

Specifies the directory in which to place the RTL Analyzer intermediate files. Allowed values are

Default value for this variable is “.".

**hdlout_internal_busses**

This variable is obsolete. This variable controls how the write -format verilog and write -format VHDL commands write out internal bused nets by parsing their names when set to true. Make sure to set `vhdlout_single_bit` and `vhdlout_preserve_hierarchical_types` to either user or vector, and `bus_inference_style` and `bus_naming_style` to the naming style (for details see the man page for `bus_inference_style` and `bus_naming_style`), when writing out VHDL files. When writing out Verilog files, set `verilogout_single_bit` to false (the default).

Default value for this variable is “false”.

---

Synthesis Variables 205
hier_dont_trace_ungroup
When false (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 thus trace between the mapped and getech (generic) circuits.

Default value for this variable is “false”.

hlo_ignore_priorities
Determines whether or not priorities set for synthetic implementations are to be observed during resource sharing and implementation selection under compile.

Default value for this variable is “false”.

hlo_minimize_tree_delay
Determines whether or not tree height minimization is enabled during resource sharing during compile, if the minimize_tree_delay attribute is not set. When true (the default value), tree height minimization is enabled on a design during resource sharing. When false, tree height minimization is disabled.

Default value for this variable is “true”.

hlo_resource_allocation
Sets the default resource sharing type to be used by compile if the resource_allocation attribute is not set. The synthetic_library variable must be set for resource sharing to take effect. Allowed values are as follows:

Default value for this variable is “constraint_driven”.
**hlo_resource_implementation**
Sets the default implementation selection type to be used by compile if the resource_implementation attribute is not set. Allowed values are as follows:

Default value for this variable is “use_fastest”.

**hlo_share_common_subexpressions**
When true (the default value), sharing of common subexpressions is enabled during resource sharing during compile. When false, sharing of common subexpressions is disabled.

Default value for this variable is “true”.

**hlo_share_effort**
Sets the relative amount of CPU time spent during resource sharing during compile. Allowed values are low, medium, or high, indicating increasing amounts of CPU time. The synthetic_library variable must be set for resource sharing to take effect.

Default value for this variable is “low”.

**hlo_transform_constant_multiplication**
When true, elaborate replaces all multiplications that have one constant input by a series of shift, add, and subtract operations. The default is false, meaning that the multiplications are not replaced.

Default value for this variable is “false”.

**insert_dft_clean_up**
When true (the default), insert_dft uses area recovery techniques to reduce the amount of test point logic. When false, insert_dft does not optimize the test point logic.

Default value for this variable is “true”.

---

Synthesis Variables 207
**insert_test_design_naming_style**

Specifies how insert_test names new designs created during the addition of test circuitry. When insert_test 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.

Default value for this variable is “%s_test_%d”.

**jtag_manufacturer_id**

The jtag_manufacturer_id variable specifies the 11-bit JEDEC code identifying the manufacturer of a design. The manufacturer_id is one of three data items permanently loaded into the JTAG device identification (ID) register (the others are the part number and the version number of the design). If an ID register is specified as part of the JTAG architecture, it is synthesized with this number in its manufacturer ID number field (bits 1 - 11).

Default value for this variable is “0”.

**jtag_part_number**

The jtag_part_number variable specifies the 16-bit part number of a design. The part number is one of three data items permanently loaded into the JTAG device identification (ID) register. The others are manufacturer_id and version number.

Default value for this variable is “65535”.

**jtag_port_drive_limit**

The jtag_port_drive_limit variable specifies the limit on the number of outputs that a JTAG Boundary Scan Register (BSR) cell can drive; the default is 6. To determine the current value of the BSR cell port drive limit, execute report_test -jtag.

Default value for this variable is “6”.

---

208 Synthesis Variables
**jtag_test_clock_port_naming_style**
Specifies the default naming convention for the TCK signal of the 1149.1 Test Access Port added to the top level of the design during boundary-scan synthesis.
Default value for this variable is “jtag_tck%s”.

**jtag_test_data_in_port_naming_style**
Specifies the default naming convention for the TDI signal of the 1149.1 Test Access Port added to the top level of the design during boundary-scan synthesis.
Default value for this variable is “jtag_tdi%s”.

**jtag_test_data_out_port_naming_style**
Specifies the default naming convention for the TDO signal of the 1149.1 Test Access Port added to the top level of the design during boundary-scan synthesis.
Default value for this variable is “jtag_tdo%s”.

**jtag_test_mode_select_port_naming_style**
Specifies the default naming convention for the TMS signal of the 1149.1 Test Access Port added to the top level of the design during boundary-scan synthesis.
Default value for this variable is “jtag_tms%s”.

**jtag_test_reset_port_naming_style**
Specifies the default naming convention for the TRST signal of the 1149.1 Test Access Port added to the top level of the design during boundary-scan synthesis.
Default value for this variable is “jtag_trst%s”.

**jtag_version_number**

The jtag_version_number variable specifies the 4-bit version number of a design. The version number is one of three data items permanently loaded into the JTAG device identification (ID) register. The others are manufacturer_id and part number.

Default value for this variable is “0”.

**lbo_cells_in_regions**

When false (the default setting), location based optimization (LBO) will put new cells at specific locations within a cluster. When set to true, LBO will convert the specific location into a preferred region for the cell. This is done by putting X_BOUNDS and Y_BOUNDS attributes on the cell when it is written to the PDEF file.

Default value for this variable is “false”.

**libgen_max_differences**

Specifies to read_lib the maximum number of differences to list between the v3.1 format description of a library cell and its statetable description. The default value, -1, allows all differences to be listed.

Default value for this variable is “-1”.

---

210 Synthesis Variables
**link_force_case**
This system variable controls the case-sensitive or insensitive behavior of the link command. Its value can be one of case_sensitive, case_insensitive, or check_reference. The default value is check_reference. The reference being linked is checked to see the case sensitivity of the input format that created that reference and that behavior is enforced. For example, a VHDL reference would be linked case-insensitively, whereas a verilog reference will be linked case-sensitively.

Default value for this variable is “check_reference”.

**link_library**
Specifies the list of design files and libraries used during linking. The link command looks at those files and tries to resolve references in the order of specified files. A * entry in the value of this variable indicates that link should 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}. You should change your_library.db to reflect your library name.

Default value for this variable is “{your_library.db}”.

**lsiin_net_name_prefix**
This variable contains the prefix added to names assigned to unnamed nets within a design read in the LSI/NDL format. Only alphanumeric characters and the underscore are allowed in the string.

Default value for this variable is “NET_”.
**lsiout_inverter_cell**

When this variable is set, the write -format lsi command does not directly connect logic zero or logic one to internal pins or ports. The purpose is to satisfy an LSI Logic design rule.

Default value for this variable is ‘”’.

**lsiout_upcase**

When this variable is set to true, all names within LSI/NDL netlists are converted to uppercase.

Default value for this variable is “true”.

**ltl_enable_mean_physical_port_location**

When true, reoptimize_design uses the arithmetic mean of all physical locations for ports that have multiple physical equivalents. When false (the default), reoptimize_design ignores nets of ports that have multiple physical equivalents.

Default value for this variable is “false”.

**mentor_bidirect_value**

Specifies the value of the property that identifies pins or primary ports of the design as being bidirectional. You can specify the name of the property using the io variable mentor_input_output_property_name.

Default value for this variable is “INOUT”.

**mentor_do_path**

Specifies a non-default directory that contains the NETED DO macros referenced by the output of write -f mentor.

Default value for this variable is ‘”’.
**mentor_input_output_property_name**
Specifies the name of the property whose value represents the direction of pins and primary ports of the design.

Default value for this variable is “PINTYPE”.

**mentor_input_value**
Specifies the value of the property that identifies input pins or primary input ports. You can specify the property name using the io variable `mentor_input_output_property_name`.

Default value for this variable is “IN”.

**mentor_logic_one_value**
Specifies the value of the property that identifies logic one nets. You can specify the name of the property using the io variable `mentor_logic_zero_one_property_name`.

Default value for this variable is “ISF”.

**mentor_logic_zero_one_property_name**
Specifies the name of the property whose value determines whether the net is connected to logic one (power) or logic zero (ground).

Default value for this variable is “INIT”.

**mentor_logic_zero_value**
Specifies the value of the property that identifies logic zero nets. You can specify the property name using the io variable `mentor_logic_zero_one_property_name`.

Default value for this variable is “OSF”.

---

Synthesis Variables 213
**mentor_output_value**
Specifies the value of the property that identifies output pins or primary output ports. You can specify the name of the property using the io variable `mentor_primitive_property_name`.

Default value for this variable is “OUT”.

**mentor_primitive_property_name**
Specifies the name of the property placed on all automatically-generated symbols.

Default value for this variable is “PRIMITIVE”.

**mentor_primitive_property_value**
Specifies the value of the property placed on all automatically-generated symbols. You can specify the name of the property using the io variable `mentor_primitive_property_name`.

Default value for this variable is “MODULE”.

**mentor_reference_property_name**
Specifies the name of the property that contains the reference name of every instance. The reference name is the name of the cell of which this object is an instantiation.

Default value for this variable is “COMP”.

**mentor_search_path**
When this variable is set, a NETED SEARCH command is added by write -f mentor to the beginning of an output file. This variable contains the list of directories to be searched for cells to be instantiated while executing the NETED DO macro.

Default value for this variable is “”. 

---

214 Synthesis Variables
mentor_write_symbols
When this variable is set to true, write -f mentor includes symbol (SYMED) information in the output file for all automatically-generated symbols.

Default value for this variable is “true”.

mgi_scratch_directory
Specifies a directory in which to store the intermediate files created by external generator(s). By default, the scratch directory is set to designware_generator in the current working directory.

Default value for this variable is “{}”.

multi_pass_test_generation
When false (the default value), every test generation/fault simulation run will be an independent run targeting the entire fault list. When true, each test generation/fault simulation run will be an incremental run targeting only the faults left undetected by the preceding test program sequence.

Default value for this variable is “false”.

pla_read_create_flip_flop
Affects read -f pla and, when set to true, enables output register information in PLA files to be read in and stored so that the output registers are instantiated within the design. Currently restricted to PLA descriptions of D flip-flops. When this variable is set to false, only combinational information is extracted from the PLA description.

Default value for this variable is “false”.

plot_box
When true, a box is drawn around the plot.

Default value for this variable is “false”.
plot_command
Specifies the operating system command that produces a hard copy of the plot. The PostScript output of the plot command is piped as standard input (stdin) to the given command.

Default value for this variable is “lpr”.

plot_orientation
Specifies if the schematic is vertical or horizontal. If landscape, the plot is output horizontally. If portrait, the plot is output vertically. With best_fit (default), the plot is output in a manner that best fits the aspect ratio of the current design schematic.

Default value for this variable is “best_fit”.

plot_scale_factor
Specifies a scaling factor for the schematic. Line widths are multiplied by the specified percentage value when the schematic is plotted.

Default value for this variable is “100”.

plotter_maxx
Specifies the x coordinate of the upper right corner of the plot output device. Usually, this variable is set only once for a site in the .synopsys_dc.setup initialization file.

Default value for this variable is “584”.

plotter_maxy
Specifies the y coordinate of the upper right corner of the plot output device. Usually, this variable is set only once for a site in the .synopsys_dc.setup initialization file.

Default value for this variable is “764”.

---

216 Synthesis Variables
plotter_minx
Specifies the x coordinate of the lower left corner of the plot output device. Usually, this variable is set only once for a site in the .synopsys_dc.setup initialization file.
Default value for this variable is “28”.

plotter_miny
Specifies the y coordinate of the lower left corner of the plot output device. Usually, this variable is set only once for a site in the .synopsys_dc.setup initialization file.
Default value for this variable is “28”.

port_complement_naming_style
Defines the convention used by compile to rename ports complemented as a result of set_boundary_optimization.
Default value for this variable is “%s_BAR”.

power_do_not_size_icg_cells
When false (the default), the compile command sizes integrated clock-gating cells in the design to correct DRC violations. This results in lower area and power when the integrated clock-gating cell is the last element in the clock tree and drives all gated registers. When this variable is true, compile does not size the cells.
Default value for this variable is “false”.

---

Synthesis Variables 217
**power_hdlc_do_not_split_cg_cells**

When false (the default), elaborate splits clock-gating cells to limit their fanout, controlled by the value specified by the `set_clock_gating_style -max_fanout` command, whose default is 128. If true, elaborate 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. elaborate does not honor the `set_clock_gating_style -max_fanout` value specified.

Default value for this variable is “true”.

**power_keep_license_after_power_commands**

When true, a DesignPower license that is checked out under `dc_shell` (DesignCompiler) remains checked out throughout the `dc_shell` session. When this variable is false (the default value), the DesignPower license will remain checked out only as long as a command is using it. At the completion of the command, the license will be released.

Default value for this variable is “false”.

**power_preserve_rtl_hier_names**

When true, HDL Compiler preserves the hierarchy information of the RTL objects in the RTL design. The `rtl2saif` command, which generates RTL forward-annotation SAIF files, needs this information. When this variable is false (the default value), `rtl2saif` cannot extract the correct synthesis invariant objects since the hierarchy information is not preserved.

Default value for this variable is “false”.
**power_rtl_saif_file**

This variable tells rtl2saif where to store the forward-annotation SAIF file if you do not specify the -output option. The default for this variable is power_rtl.saif.

Default value for this variable is “power_rtl.saif”.

**power_sdpd_saif_file**

This variable tells lib2saif where to store the forward-annotation SAIF file if you do not specify the -output option. The default for this variable is power_sdpd.saif.

Default value for this variable is "power_sdpd.saif".

**read_db_lib_warnings**

When true, indicates that warnings are to be printed while a technology db library is being read in with read. When false (the default), no warnings are given.

Default value for this variable is “false”.

**read_name_mapping_nowarn_libraries**

Specifies a list of libraries for which no warning messages are to be issued by read -f edif -names_file if the libraries are not found. The default is to issue warning messages for all libraries not found. For example, if the library foo does not exist and read_name_mapping_nowarn_libraries is set to its default value, warning messages are issued saying that the library foo has not been found. However, if read_name_mapping_nowarn_libraries = {foo}, then the warning messages are suppressed. These warning (error) messages include all components in that library and all pins in those components.

Default value for this variable is “{"}”.
**read_translate_msff**
When true (the default), indicates that master-slave flip-flops (that is, those specified with the clocked_on_also syntax) are to be automatically translated to master-slave latches. When false, both master and slave remain flip-flops.

Default value for this variable is “false”.

**reoptimize_design_changed_list_file_name**
When the reoptimize_design_changed_list_file_name variable is set to a particular file name, the reoptimize_design -in_place and reoptimize_design -post_layout_opto commands will output a list of design modifications/additions to the file. The commands will use the file to store the list of those cells which changed and those cells and nets which were added. By default, this variable is set to the empty string, and no such file is created during post_layout or in_place optimization.

Default value for this variable is “file_name”.

**rtl_load_resistance_factor**
Used with the set_rtl_load command to specify default resistance values for rtl loads.

Default value for this variable is “0.0”.

**sdfin_fall_cell_delay_type**
Specifies the delay type for fall cell delays read from a timing file in SDF format. Delays from timing files are annotated in Design Compiler with read_timing. The delay type can be minimum, typical, or maximum. Timing files in SDF format can contain minimum, typical, and maximum delay values, but Design Compiler can annotate only one type on the current design.

Default value for this variable is “maximum”.
**sdfin_fall_net_delay_type**

Specifies the delay type for fall net delays read from a timing file in SDF format. Delays from timing files are annotated in Design Compiler with read_timing. The delay type can be minimum, typical, or maximum. Timing files in SDF format can contain minimum, typical, and maximum delay values, but Design Compiler can annotate only one type on the current design.

Default value for this variable is “maximum”.

**sdfin_min_fall_cell_delay**

Specifies the minimum fall cell delay read from a timing file in S.D.F. format. Delays from timing files are annotated in Design Compiler with read_timing. By default, timing values less or equal to 0 are not read. This variable must be a positive number. The unit must be the same as the timing unit in the technology library.

Default value for this variable is “0”.

**sdfin_min_fall_net_delay**

Specifies the minimum fall net delay read from a timing file in S.D.F. format. Delays from timing files are annotated in Design Compiler with read_timing. By default, timing values less or equal to 0 are not read. This variable must be a positive number. The unit must be the same as the timing unit in the technology library.

Default value for this variable is “0”.

---

Synthesis Variables 221
**sdfin_min_rise_cell_delay**
Specifies the minimum rise cell delay read from a timing file in S.D.F. format. Delays from timing files are annotated in Design Compiler with read_timing. By default, timing values less or equal to 0 are not read. This variable must be a positive number. The unit must be the same as the timing unit in the technology library.

Default value for this variable is “0”.

**sdfin_min_rise_net_delay**
Specifies the minimum rise net delay read from a timing file in S.D.F. format. Delays from timing files are annotated in Design Compiler with read_timing. By default, timing values less or equal to 0 are not read. This variable must be a positive number. The unit must be the same as the timing unit in the technology library.

Default value for this variable is “0”.

**sdfin_rise_cell_delay_type**
Specifies the delay type for rise cell delays read from a timing file in SDF format. Delays from timing files are annotated in Design Compiler with read_timing. The delay type can be minimum, typical, or maximum. Timing files in SDF format can contain minimum, typical, and maximum delay values, but Design Compiler can annotate only one type on the current design.

Default value for this variable is “maximum”.

---

222 Synthesis Variables
**sdfin_rise_net_delay_type**

Specifies the delay type for rise net delays read from a timing file in SDF format. Delays from timing files are annotated in Design Compiler with `read_timing`. The delay type can be minimum, typical, or maximum. Timing files in SDF format can contain minimum, typical, and maximum delay values, but Design Compiler can annotate only one type on the current design.

Default value for this variable is “maximum”.

**sdfin_top_instance_name**

Specifies the name prepended to all instance names in timing files in S.D.F. format. Delays from timing files are annotated in Design Compiler with `read_timing`. Set this variable when the cell instance names contain a prepended name different than the empty string.

Default value for this variable is “”.

**sdfout_allow_non_positive_constraints**

When true, `write_constraints -format sdf` can write 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.

Default value for this variable is “false”.

**sdfout_min_fall_cell_delay**

Specifies the minimum non-back-annotated fall 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.

Default value for this variable is “0”.

---

*Synthesis Variables* 223
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. 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.

Default value for this variable is “0”.

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. 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.

Default value for this variable is “0”.

sdfout_min_rise_net_delay
Specifies the minimum non-back-annotated rise 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.

Default value for this variable is “0”.
**sdfout_time_scale**

Specifies the time scale of the delays written to timing files in S.D.F. format. Delays from Design Compiler are written to timing files with write_timing. This 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 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.

Default value for this variable is “1”.

**sdfout_top_instance_name**

Specifies the name prepended to all instance names when writing timing files in S.D.F. format. Timing files are written with the command write_timing. 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.

Default value for this variable is “”.

**sdfout_write_to_output**

Specifies whether interconnect delays between cells and top level output ports will be written by write_timing -f sdf. Also determines whether output to output pin IOPATH statements will be written for cells which contain output to output timing arcs. Output to output timing is not supported in v1.0 SDF for either IOPATH or INTERCONNECT statements. However, the Synopsys Simulator does support it.

Default value for this variable is “false”.

---

*Synthesis Variables* 225
search_path
Specifies directories searched by the design and test compilers for files specified without directory names. This includes looking for technology and symbol libraries, design files, etc. This variable is a list of directory names and is usually set to a central library directory.

Default value for this variable is “{.synopsys_root + "/libraries”}”.

command_log_file
This variable is for use in dc_shell-t (Tcl mode of dc_shell) only.

Default value for this variable is “./command.log”.

sh_new_variable_message
This variable is for use in dc_shell-t (Tcl mode of dc_shell) only.

Default value for this variable is “true”.

sh_source_uses_search_path
This variable is for use in dc_shell-t (Tcl mode of dc_shell) only.

Default value for this variable is “true”.

single_group_per_sheet
Partitioning option that, when true, specifies that only one logic group is put on a sheet. This eliminates the possibility of more than one off-sheet connector with the same name on a single sheet.

Default value for this variable is “false”.

site_info_file
Contains the path to the site information file for licensing. The default is the empty string.

Default value for this variable is “”. 

226 Synthesis Variables
sort_outputs
Sorts output ports on the schematic by port name.
Default value for this variable is “false”.

suppress_errors
Specifies a list of error codes for which messages are to be suppressed during the current Design Analyzer/dc_shell session. Default is set to no error message being suppressed.
Default value for this variable is “{}”.

symbol_library
Specifies the symbol libraries to use during schematic generation. This variable is a list of symbol library names.
Default value for this variable is “{sym_lib.sdb}”.

synlib_disable_limited_licenses
When this variable is true, limited licenses for synthetic library parts will not be considered. That is, no limited licenses will be automatically checked out, and limited licenses will not be able to enable synthetic library parts.
Default value for this variable is “false”.
**synlib_dont_get_license**

Specifies a list of synthetic library part licenses that are not automatically checked out. By default, all synthetic library part licenses are automatically checked out if there is a possibility that they will be used. Add licenses to this list if they should not be automatically checked out when only the possibility of their use exists. The exception is that licenses on this list are checked out only when they are required to build an instantiated part, or when manually requested via the get_license command.

Default value for this variable is “{ }”.

**synlib_evaluation_mode**

When no Designware-Basic or Designware-FPGA-Basic keys are available, you can still evaluate synthetic library parts by setting this variable to true. When this variable is set to true, you will be authorized for the same capabilities that Designware-Basic would provide, except that all DesignWare parts and the level of hierarchy that contains them will automatically be given a limited license. Thus, you will not be able to write out any DesignWare part (including standard parts) or any design that contains a DesignWare part. You will still be able to use report_timing and report_resources to print reports that show timing and implementation selections, respectively.

Default value for this variable is “false”.
**synlib_model_map_effort**

The `synlib_model_map_effort` variable determines the `map_effort` used during modeling of synthetic library parts. Allowable values for this variable are low, medium, high, 1, 2, and 3. The default value is medium. This variable can be overridden on the current design using the `set_model_map_effort` command.

Default value for this variable is “medium”.

**synlib_optimize_non_cache_elements**

If a synthetic library model cannot be retrieved from the cache, then the part description must be created and, possibly, optimized. This variable controls whether non-cached models are optimized or used in an unoptimized form. When the variable is true, either optimized models will be read from the cache, or part descriptions will be read in and optimized. When false, either optimized models will be read from the cache, or part descriptions will be created and used as unoptimized netlists. Using unoptimized models will decrease the quality of results of TDRS and implementation selection.

Default value for this variable is “true”.
**synlib_prefer Ultra_license**

When true, any use of Foundation library parts sets the Design Compiler ultra optimization mode if it is not already set, and uses the DesignWare-Foundation-Ultra license instead of the DesignWare-Foundation license. When false (the default), if Design Compiler is in the ultra optimization mode, Foundation parts use the DesignWare-Foundation-Ultra license. If Design Compiler is not in the ultra optimization mode, Foundation parts use the DesignWare-Foundation license.

Default value for this variable is “false”.

**synlib_sequential_module**

Controls the amount of processing to be done during resource sharing and implementation selection by compile on synthetic library modules that have implementations with sequential elements. Allowed values are default (the default), iis_processing, one_implementation_choice, and multiple_implementation_choices. These values are described below.

Default value for this variable is “default”.
**synlib_wait_for_design_license**

Specifies a list of authorized synthetic library licenses that are to be waited for. By default, Design Compiler checks for all design licenses and checks out one that is available. If none of the licenses are available, Design Compiler terminates the command that requires the license. You can override this default behavior by setting this variable with a list of licenses. When this variable is set, if no licenses are available, Design Compiler does not terminate the command that requires the license; instead, it waits for the listed licenses to become available.

Default value for this variable is “{}”.

**syntax_check_status**

One of a pair of status variables, syntax_check_status and context_check_status, whose values are set by the Syntax Checker and not by the user. You examine these variables to determine the status of the syntax_check or context_check mode of the Syntax Checker. A value of true indicates that the mode is currently enabled; a value of false indicates that the mode is disabled. For example, a value of syntax_check_status = true indicates that the syntax_check mode is currently enabled. Both modes cannot be enabled simultaneously; these two status variables allow you to determine whether one mode is enabled before you attempt to enable the other mode.

Default value for this variable is “variable”.

**synthetic_library**
Specifies a list of synthetic libraries to use when compiling. Default is `{ }`, synthetic_library works much like the target_library variable does for technology libraries. Synthetic_library can be set to be a list of zero or more sldb files that you wish to use in the compile or replace_synthetic commands. When synthetic operators and/or modules are processed in compile, the operators, bindings, modules, and implementations of the specified library(s) 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.

Default value for this variable is “{ }”.

**target_library**
Specifies the list of technology libraries of components used when compiling a design. Default is `{your_library.db}`. You should change this to reflect your library name.

Default value for this variable is “{your_library.db}”.

**tdlout_upcase**
When this variable is set to true, all names within TDL net lists are converted to upper case.

Default value for this variable is “true”.

**template_naming_style**
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. When a module is built, a unique name is automatically generated using the module name, parameter names, and parameter values.

Default value for this variable is “%s_%p”.

---

232 Synthesis Variables
**template_parameter_style**
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. When a module is built, a unique name is automatically generated using the module name, parameter names, and parameter values.

Default value for this variable is “%s%d”.

**template_separator_style**
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. When a module is built, a unique name is automatically generated using the module name, parameter names, and parameter values.

Default value for this variable is “_”.

**test_allow_clock_reconvergence**
When true (the default), check_test allows reconvergent nets that originate from the same top-level clock port; that is, it does not generate an X when multiple edges reconverge at a net, unless the edges actually conflict. Notice that an X is still generated when multiple edges reconverge at a gate, and for all reconvergent edges that originate from different top-level clock ports.

Default value for this variable is “true”.
**test_bsd_allow_tolerable_violations**
When true, allows optimize_bsd to replace observe_and_control BSR cells with observe_only cells, or to remove the BSR cells altogether during timing-driven or area-driven optimization. This can give rise to tolerable violation during execution of the check_bsd command. When false (the default), cells cannot be replaced.

Default value for this variable is “FALSE”.

**test_bsd_control_cell_drive_limit**
Specifies the number of cells that can be driven by a single BSR control cell during optimize_bsd while optimizing control cell allocation. The test_bsd_optimize_control_cell variable must be set to true for the optimization to take place. If area constraints are not violated, the optimization does not take place. You assign area constraints to the current design by using set_max_area.

Default value for this variable is “3”.

**test_bsd_manufacturer_id**
Specifies the manufacturer id to be used to create the value captured in the Device Identification Register during execution of the insert_bsd command. This represents the manufacturer id assigned to the organization and is unique.

Default value for this variable is “0”.

---

234 Synthesis Variables
**test_bsd_optimize_control_cell**

When true, allows optimize_bsd to optimize allocation of 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 will not take place. You use set_max_area to specify area constraints for the current design.

Default value for this variable is “FALSE”.

**test_bsd_part_number**

Specifies the part number to be used to create the value captured in the Device Identification Register during execution of the insert_bsd command. This represents the part number assigned to the IC and should be unique inside the organization.

Default value for this variable is “0”.

**test_bsd_version_number**

Specifies the version number to be used to create the value captured in the Device Identification Register during execution of the insert_bsd command. This represents the version number assigned to this IC (Integrated Circuit) and is unique for this IC.

Default value for this variable is “0”.

**test_bsdl_default_suffix_name**

Specifies the default suffix for the name of the BSDL file generated by the write_bsdl command. If no output file name is specified by the write_bsdl command, by default, the output file is named top_level_design_name.suffix, where suffix is the value of test_bsdl_default_suffix_name.

Default value for this variable is “bsdl”.

---

Synthesis Variables  235
test_bsdl_max_line_length
Specifies the maximum number of characters per line for the output BSDL file produced by the write_bsdl command. The maximum line length cannot be less than the default of 80; if you specify a number lower than the default, write_bsdl uses the default instead.

Default value for this variable is “74”.

test_capture_clock_skew
Allows you to specify a qualitative measure of clock skew. Values are no_skew, small_skew (the default), or large_skew, defined in later paragraphs.

Default value for this variable is “small_skew”.

test_cc_ir_masked_bits
An integer whose binary value identifies instruction register (IR) bits that are to be masked during the search for all possible implemented instructions by check_bsd. By default, no bits are masked.

Default value for this variable is “0”.

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 for all possible implemented instructions by check_bsd. Bits to be masked are identified by the value of the test_cc_ir_masked_bits variable. By default, zeros are forced into all masked bits.

Default value for this variable is “0”.
test_check_port_changes_in_capture
When true (the default), check_test checks for changes in values applied to bidirectional ports in the parallel measure cycle, which can cause an unreliable capture in sequential cells. When false, these checks are disabled.

Default value for this variable is “true”.

test_clock_port_naming_style
Specifies the naming style used by insert_test for global 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_scan_signal or set_signal_type command.)

Default value for this variable is “test_c%s”.

test_dedicated_subdesign_scan_outs
When true, makes Test Compiler create dedicated scan-out ports on subdesigns. When false (the default), Test Compiler uses existing subdesign ports where possible.

Default value for this variable is “FALSE”.

test_default_bidir_delay
A positive real number in nanoseconds, which defines the default switching time of bidirectional ports in a tester cycle. This variable is used by both ATPG and TestSim.

Default value for this variable is “55.0”.

---

Synthesis Variables 237
**test_default_client_order**

Enables or disables test point utilities. Allowed values are \{\} (the default), which disables all test point utilities; autofix, which enables the Autofix utility; and wrapper, which enables the wrapper utility. When Autofix and wrapper are disabled, insert_dft performs only scan insertion and routing. You can specify either autofix, wrapper or both; the order determines the order in which they are invoked by insert_dft. To use any of the Autofix or wrapper commands, you must first enable the appropriate utility using this variable.

Default value for this variable is “\{\}”.

**test_default_delay**

A positive real number in nanoseconds, which defines the default time in a tester cycle to apply values to input ports. This variable is used by both ATPG and TestSim.

Default value for this variable is “5.0”.

**test_default_min_fault_coverage**

Specifies the default desired minimum fault coverage percent for the current design for the partial scan test methodology when the set_min_fault_coverage command has not be used for the current design.

Default value for this variable is = “95”.
test_default_period
Defines the default length of a test vector cycle. This value translates directly to the speed of application of the test vectors on ATE. For example, ATE running at 10 MHz would imply a period of 100 ns. Changing the value of this variable will modify the content of the inferred test protocol generated by the check_test command. The user can overwrite the value of this variable by providing a user-defined test protocol and read it in using the read_test_protocol command or using the create_test_clock command. The write_test command uses the value of this variable stated in the test protocol attached to the current design to generate the test program.

Default value for this variable is “100.0”.

test_default_scan_style
Defines the default insert_test scan style to use if a scan style is not specified using set_scan_style. The variable must identify one type of a supported scan style, currently multiplexed_flip_flop, clocked_scan, lssd, aux_clock_lssd, combinational, or none. The default is multiplexed_flip_flop.

Default value for this variable is “multiplexed_flip_flop”.

test_default_strobe
A positive real number in nanoseconds, which defines the default strobe time in a tester cycle for output and bidirectional ports in output mode. This variable is used by both ATPG and TestSim.

Default value for this variable is “95.0”.
**test_default_strobe_width**

Defines the default strobe pulse width, i.e. the default time that specifies how long the strobe pulse needs to be held active after invocation. Changing the value of this variable will modify the content of the inferred test protocol generated by the check_test command. The user can overwrite the value of this variable by providing a user-defined test protocol and read it in using the read_test_protocol command. The write_test command uses the value of this variable stated in the test protocol attached to the current 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 will simply be ignored.

Default value for this variable is “10.0”.

**test_design_analyzer_uses_insert_scan**

When true (the default), design_analyzer executes insert_scan when the Tools/Test Synthesis.../Insert Internal Scan Circuitry... menu is selected in the Design Analyzer text window and the OK button is pushed. When false, design_analyzer executes insert_test instead of insert_scan.

Default value for this variable is “true”.

**test_disable_find_best_scan_out**

When false (the default), insert_scan and insert_test select the scan-out pin on a scan cell that has the greatest timing slack. Typically, this means that insert_scan and insert_test make extensive use of Qbar pins to drive the scan-in pins of scan cells. Test Compiler supports inversions on the scan chain.

Default value for this variable is “FALSE”.
**test_dont_fix_constraint_violations**
When false (the default), insert_scan and insert_test attempt to minimize performance constraint violations. You can disable this behavior by setting the variable to true.

Default value for this variable is “false”.

**test_infer_slave_clock_pulse_after_capture**
Allows you to guide protocol inference in check_test for master/slave-based test design methodologies. When set to infer (the default value), check_test protocol inference will be based on an analysis of the scan cell states instead of the scan style as it was previously done (1997.01 and earlier releases). The other values are pulse and no_pulse. If you set the variable to pulse, all slave clocks are pulsed after capture. If you set it 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 variable value inference is taken by default.

Default value for this variable is “infer”.

**test_isolate_hier_scan_out**
When 1, insert_test inserts logic that isolates scan connections at hierarchical boundaries during functional operation; this can reduce dynamic switching currents and output loading. When 0 (the default), no logic is inserted.

Default value for this variable is “0”.

---

Synthesis Variables 241
**test_jump_over_bufs_invs**
A boolean variable, which specifies if the internal clock detection mechanism considers the output pins of buffers and inverters as internal clocks. The default value is true. Only multiple-input gates output pins are considered as internal clocks.

Default value for this variable is “TRUE”.

**test_mode_port_inverted_naming_style**
Specifies the naming style used for the type test_hold_logic_zero test mode signal ports created in designs. (New ports are not added if suitable ports are identified with the set_test_mode_signal command.)

Default value for this variable is “test_mode_i%s”.

**test_mode_port_naming_style**
Specifies the naming style used for the type test_hold_logic_one test mode signal ports created in designs. (New ports are not added if suitable ports are identified with the set_test_mode_signal command.)

Default value for this variable is “test_mode%s”.

**test_non_scan_clock_port_naming_style**
Specifies the naming style used by insert_test for ports created by clock gating for non-scan clocks are to be named. The %s is filled with a string identifying the original clock, its inversion (either i for inverted or n for not-inverted), and inactive level (either 0 or 1).

Default value for this variable is “test_nsc_%s”.
test_point_keep_hierarchy
When false (the default), insert_dft synthesizes test points and ungroups the test point design. When true, insert_dft keeps each test point design in a separate level of hierarchy. As a consequence, this also prevents insert_dft from optimizing the test point logic.

Default value for this variable is “false”.

test_preview_scan_shows_cell_types
Tells preview_scan to show cell instance types. By default, cell type information is suppressed.

Default value for this variable is “false”.

test_protocol_add_cycle
When true (the default), if you specify design bidirectional ports as inputs during the scan shift, check_test adds an extra cycle after the shift cycle in the test protocol. In the extra cycle, all bidirectional ports are set to output mode. If you do not want check_test to add the extra cycle, set this variable to false.

Default value for this variable is “true”.

test_scan_clock_a_port_naming_style
Used the same as test_clock_port_naming_style.

Default value for this variable is “test_sca%s”.

test_scan_clock_b_port_naming_style
Used the same as test_clock_port_naming_style.

Default value for this variable is “test_scb%s”.

test_scan_clock_port_naming_style
Used the same as test_clock_port_naming_style.

Default value for this variable is “test_scs%s”.
**test_scan_enable_inverted_port_naming_style**
Used the same as `test_clock_port_naming_style`.
Default value for this variable is “test_sei%s”.

**test_scan_enable_port_naming_style**
Used the same as `test_clock_port_naming_style`.
Default value for this variable is “test_se%s”.

**test_scan_in_port_naming_style**
Specifies the naming style used by `insert_test` 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_scan_signal` or `set_signal_type` command.)
Default value for this variable is “test_si%s%s”.

**test_scan_link_so_lockup_key**
Tells `preview_scan` what key to use to identify cells with scan-out lockup latches in reports. By default, l is used.
Default value for this variable is “l”.

**test_scan_link_wire_key**
Tells `preview_scan` what key to use to identify cells that drive wire scan links in reports. By default, w is used.
Default value for this variable is “w”.

**test_scan_out_port_naming_style**
Used the same as `test_scan_in_port_naming_style`.
Default value for this variable is “test_so%s%s”.

**test_scan_segment_key**
Tells `preview_scan` what key to use to identify scan segments in reports. By default, s is used.
Default value for this variable is “s”.
**test_scan_true_key**

Tells preview_scan what key to use to identify cells with true scan attributes in reports. By default, t is used.

Default value for this variable is “t”.

**test_stil_multiclock_capture_procedures**

When true, the write_test_protocol -format stil command creates capture procedures in the STIL protocol with multiple clocks active in each procedure. The groups of clocks (capture clock groups) are determined by check_test. When false (the default), write_test_protocol places only one clock pulse in each capture clock procedure.

Default value for this variable is “FALSE”.

**test_stil_netlist_format**

Specifies the netlist format to be used by the write_test_protocol command when writing out STIL protocol files. Allowed values are db (the default), verilog, or vhdl. Set this variable to match the netlist format of your design before issuing write_test_protocol, so that the port names in the STIL protocol file match the port names in the netlist.

Default value for this variable is “db”.

**test_user_defined_instruction_naming_style**

 Specifies the naming style used by check_bsd and write_bsd for the user-defined (non-standard) 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 are USER1, USER2, etc.

Default value for this variable is “USER%d”.

---

Synthesis Variables 245
test_user_test_data_register_naming_style

Specifies the naming style used by check_bsd and write_bsd for the user-defined (non-standard) 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 are UTDR1, UTDR2, etc.

Default value for this variable is “UTDR%d”.

test_write_four_cycle_stil_protocol

When true, the write_test_protocol -format stil command inserts in the output STIL protocol file a dummy cycle between all measure and capture cycles in the STIL protocol. When false (the default), no additional cycle is inserted.

Default value for this variable is “FALSE”.

testsim_print_stats_file

Note: TestSim is obsolete with 1999.10, and has been replaced by the TetraMax fault simulator. For more information, see the TetraMax documentation.

text_editor_command

Specifies the command that executes when the Edit/File menu is selected in the Design Analyzer text window.

Default value for this variable is “xterm”.

text_print_command

Specifies the command that executes when the File/Print menu is selected in the Design Analyzer text window.

Default value for this variable is “lpr”.

---

246 Synthesis Variables
**timing_self_loops_no_skew**
This Boolean variable affects the behavior, runtime, and cpu usage of report_timing and compile. When true, clock skew is eliminated for a path starting and ending at the same register. When 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 the variable to true, but note that runtime and memory usage may increase significantly.

Default value for this variable is “false”.

**true_delay_prove_false_backtrack_limit**
An integer variable that specifies the number of backtracks to be used by report_timing -true in searching for false paths; -1 specifies unlimited backtracking. The default is 1000. One of a pair of variables that includes true_delay_prove_true_backtrack_limit.

Default value for this variable is “1000”.

**true_delay_prove_true_backtrack_limit**
An integer variable that specifies the number of backtracks to be used by report_timing -true in searching for true paths; -1 specifies unlimited backtracking. The default is 1000. One of a pair of variables that includes true_delay_prove_false_backtrack_limit.

Default value for this variable is “1000”.
**uniquify_naming_style**

Specifies the naming convention used by uniquify. The variable string must contain only one %s and %d character sequence. To use a percent sign in the design name, two are needed in the string (%%).

Default value for this variable is “%s_%d”.

**use_port_name_for_oscs**

Partitioning option that, when true, specifies that off-sheet connectors for nets that also have ports on them are given the name of the port. When false, the connectors are given the name of the net.

Default value for this variable is “true”.

**verbose_messages**

When true, causes more explicit system messages to be displayed during the current Design Analyzer dc_shell session. The default is true.

Default value for this variable is “true”.

**verilogout_debug_mode**

This variable is to be used only when the variable verilogout_levelize is set to true.

Default value for this variable is “false”.

**verilogout_equation**

When true, writes Verilog assign statements (Boolean equations) for combinational gates, rather than gate instantiations.

Default value for this variable is “false”. 
verilogout_higher_designs_first
When true, writes Verilog modules ordered so that higher level designs come before lower level designs as defined by the design hierarchy. Default is to write lower level designs first.

Default value for this variable is “false”.

verilogout_ignore_case
When true, case is not considered when identifiers are compared against Verilog reserved words.

Default value for this variable is “false”.

verilogout_include_files
When true and verilogout_levelize is true, write -f verilog writes an include statement with the value that you set in this variable. For example, when verilogout_include_files={my_header.v}, you see an include my_header.v in your verilog output.

Default value for this variable is “{ }”.

verilogout_levelize
When true, write -f verilog levelizes and flattens the netlist and, in addition, replaces standard DesignWare operations with simulatable Verilog before writing out the netlist. For a design that has been run through Behavioral Compiler, but has not yet been compiled, you must set this variable to true. When false (the default), the netlist is neither levelized nor flattened; moreover, the DesignWare operations are written out as black boxes, so that the Verilog written out cannot be simulated.

Default value for this variable is “false”.
verilogout_no_negative_index
When true, shifts the negative range to the positive range starting at 0. For example, if you have 0 downto -7, it becomes 7 downto 0.

Default value for this variable is “false”.

verilogout_no_tri
When true, three-state nets are declared as Verilog wire instead of tri. This variable is useful in eliminating assign primitives and tran gates in the Verilog output.

Default value for this variable is “false”.

verilogout_show_unconnected_pins
When this variable is set to true, the verilog writer in dc_shell writes all the unconnected instance pins when connecting module ports by name. For example, modb b1 (.A(in),.Q(out),.Qn()).

Default value for this variable is “false”.

verilogout_single_bit
When true, does not output vectored ports in Verilog output. Instead, all vectors are written as single bits.

Default value for this variable is “false”.

verilogout_unconnected_prefix
When this variable is set, the verilog writer in dc_shell uses this name to create unconnected wire names. The general form of the name is SYNOPSYS_UNCONNECTED_%d.

Default value for this variable is “SYNOPSYS_UNCONNECTED_”.
**vhdl_lib_architecture**

Determine the VHDL model types for write_lib to generate. UDSM selects Unit-Delay Structural Models; FTSM selects Full-Timing Structural Models; FTGS selects Full-Timing Gate-Level Simulation models; and VITAL selects VITAL Simulation Models.

Default value for this variable is “{UDSM, FTSM, FTGS, VITAL}”.

**vhdl_lib_glitch_handle**

Determines if timing hazards have glitch-forced (on-detect) or spike-forced (on-event) Xs. When true (the default value), Xs are glitch-forced; when false, spike-forced.

Default value for this variable is “true”.

**vhdl_lib_logic_system**

Selects the logic system in which to create the VHDL libraries. Currently, only ieee-1164 for the IEEE 1164.1 nine value (std_logic) logic system is allowed.

Do not change this variable's value to anything other than ieee-1164.

Default value for this variable is “ieee-1164”.

**vhdl_lib_logical_name**

This variable defines the logical name to be used by the VHDL libraries. If the variable is set to an empty string, then the file basename is used as the default.

Default value for this variable is “”.
**vhdllib_negative_constraint**

Determines whether or not a generated VITAL model is to have negative constraint handling capability. When true, all cells in the generated VITAL library can handle negative timing constraints, ignoring the cell attribute handle_negative_constraint value set in the technology library. When false (the default), only cells in the generated VITAL library whose cell attribute handle_negative_constraint is true can handle negative timing constraints. vhdllib_negative_constraint does not affect any simulation models other than VITAL.

Default value for this variable is “FALSE”.

**vhdllib_pulse_handle**

Determines the algorithm used to handle timing hazards for FTGS model. Values are glitch, spike, transport, inertial, or use_vhdllib_glitch_handle (the default). The default causes the hazard handling algorithm to be selected by the variable vhdllib_glitch_handle. vhdllib_pulse_handle does not affect any simulation models other than FTGS.

Default value for this variable is “use_vhdllib_glitch_handle”.

**vhdllib_tb_compare**

Controls library testbench generation. No testbenches are created if this variable is set to zero. Otherwise, vhdllib_tb_compare specifies the default value for the testbenches Cmp_Algorithm generic Integer flag. The greater the number from 1 to 5, the more rigorous the verification.

Default value for this variable is “0”.
**vhdl_lib_tb_x_eq_dontcare**

Specifies the default value for the testbenches X_Eq_DontCare generic Boolean flag. If X_Eq_DontCare is true, then X states are ignored during output comparisons.

Default value for this variable is “true”.

**vhdl_lib_timing_checks**

Determines the default value for the cell TimingChecksOn generic Boolean flag in the VITAL model.

Default value for this variable is “true”.

**vhdl_lib_timing_mesg**

Determines the default value for the cell Timing_mesg generic Boolean flag in FTSM, UDSM, and FTGS models. It also determines the value of the GlitchMode parameter for the VitalPropagatePathDelay procedure in the VITAL model.

Default value for this variable is “true”.

**vhdl_lib_timing_xgen**

Determines the default value for the cell Timing_xgen generic Boolean flag in FTSM, UDSM, and FTGS models. It also determines the default value for the cell XGenerationOn generic Boolean flag in the VITAL model.

Default value for this variable is “false”.

---

*Synthesis Variables 253*
**vhdlout_architecture_name**
This variable determines the name to be used for the architectures written out by write -f vhdl. Legal characters are letters, digits, and underscores. The strings %a and %u are replaced by the original architecture name and a unique ID, respectively. (%a and %A are synonyms, as are %u and %U).
Default value for this variable is “SYN_%a_%u”.

**vhdlout_bit_type**
Sets the basic bit type in a design written to VHDL. This is useful when your design methodology is based on some other logic value system than std_logic.
Default is std_logic.
Default value for this variable is “std_logic”.

**vhdlout_bit_type_resolved**
When TRUE (the default value), prevents VHDLout from creating new bus resolution functions when writing wired logic. When FALSE, VHDLout creates bus resolution functions for signals with more than one driver. Set this variable to TRUE if the vhdlout_bit_type is a resolved type.
Default value for this variable is “TRUE”.

**vhdlout_bit_vector_type**
Sets the basic bit_vector type in a design written to VHDL. This is useful when your design methodology is based on some other logic value system than std_logic.
Default value for this variable is “std_logic_vector”.

---

254 Synthesis Variables
vhdlout_conversion_functions

Overrides conversion functions that are written out. The value of this variable is a list of lists. Each lower level list is a list of three strings representing the from type, the to type, and the function used for the conversion. Use base types in the from and to type fields.

Default value for this variable is “{}”.

vhdlout_debug_mode

This variable is to be used only when the variable vhdlout_levelize is set to true.

Default value for this variable is “false”.

vhdlout_dont_create_dummy_nets

When true, the vhdl writer does not create any dummy nets for connecting unused pins or ports.

Default value for this variable is “false”.

vhdlout_dont_write_types

When true, and if vhdlout_single_bit is false, type declarations for any types that are declared in the original VHDL are not written. Only special types needed by vhdlout are declared and are given unique names. Other types should be declared in a user-defined package using vhdlout_use_packages.

Default value for this variable is “false”.

vhdlout_equations

When true, combinational logic is written as technology-independent Boolean equations, and sequential logic is written as technology-independent wait statements. When false (default), all logic is written as technology-specific netlists.

Default value for this variable is “false”.

---

Synthesis Variables
vhdlout_follow_vector_direction
When vhdlout_single_bit is set to vector, the array always uses ascending range, regardless of the original array range direction. Use this variable to reflect the original array type direction.

Default value for this variable is “false”.

vhdlout_levelize
For a design that has been run through Behavioral Compiler, when this variable is true, write `-f vhdl levelizes and flattens a netlist before writing it out. When false (the default), the netlist written out is neither levelized nor flattened, but contains concurrent assignments.

Default value for this variable is “false”.

vhdlout_one_name
Determines the literal name for 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. This is used with vhdlout_bit_type. Default is `1'.

Default value for this variable is “1”.

vhdlout_package_naming_style
This variable determines the name that will be used for the type conversion packages written out by vhdlout. It may contain a string made up of letters, digits and underscores. The string %d will be replaced by the name of the ’current_design’. (%d and %D are synonyms.) This way all conversion packages that are written out are guaranteed to get unique names.

Default value for this variable is “CONV_PACK_%d;”.
vhdlout_preserve_hierarchical_types
This variable affects how ports on lower-level designs are written out with write -f vhdl. A lower-level design is instantiated by any of the designs being written out. In contrast, ports on top-level designs are controlled by vhdlout_single_bit.
Default value for this variable is “VECTOR”

vhdlout_separate_scan_in
This variable affects how the scan chain is written out in VHDL. If this variable is set to FALSE, then the scan chain is written out in the same file as the design. The scan chain is not visible in the test bench and, therefore, parallel-loading is not possible.
Default value for this variable is “FALSE”.

vhdlout_single_bit
This variable affects how ports on the top-level design are written out and can have values USER (FALSE), VECTOR, or BIT (TRUE). Lower level design ports are controlled by vhdlout_preserve_hierarchical_types. (A design is considered lower level if it is instantiated by any of the designs being written out.)
Default value for this variable is “USER”.

vhdlout_synthesis_off
This variable has an effect only if vhdlout_write_top_configuration is true.
Default value for this variable is “true”.

vhdlout_target_simulator
Names the target simulator to which the VHDL file is written. Currently, the only valid value is xp.
Default value for this variable is “”.
**vhdlout_three_state_name**
Names the high impedance bit value used for three-state device values.
Default value for this variable is “Z”.

**vhdlout_three_state_res_func**
Names a user-supplied three_state resolution function, which must bin one of the packages specified by vhdlout_use_packages. If this variable is set to an empty string, a default three-state resolution function is written out, if needed. The default three-state resolution function drives a signal to unknown if the signal is driven more than once by logic zero or logic one.
Default value for this variable is “”.

**vhdlout_time_scale**
Specifies the scaling of the delays written to timing files in Synopsys VHDL format. Delays from Design Compiler are written to timing files with the write_timing command. This 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. The time unit of delays in the timing file in Synopsys VHDL format is always 1 nanosecond.
Default value for this variable is “1”.

**vhdlout_top_configuration_arch_name**
Depending on the setting of vhdlout_write_top_configuration, a configuration statement is written out by VHDLout.
Default value for this variable is “A”.

---

258 Synthesis Variables
**vhdlout_top_configuration_entity_name**
Depending on the setting of `vhdlout_write_top_configuration`, a configuration statement is written by VHDLout.
Default value for this variable is “E”.

**vhdlout_top_configuration_name**
This variable determines the name of the configuration that is written out under control of `vhdlout_write_top_configuration`.
Default value for this variable is “CFG_TB_E”.

**vhdlout_unknown_name**
Specifies the value used to drive a signal to the unknown state. The default for this variable is 'X'.
Default value for this variable is “X”.

**vhdlout_upcase**
When true, the VHDL writer converts everything to uppercase before writing out the netlist. The default is false.
Default value for this variable is “false”.

**vhdlout_use_packages**
List of package names. A use clause is written into the VHDL file for each of these packages for all entities. library clauses are also written out as needed. If this variable is set to an empty list ({}), it has no effect on the write command.
Default value for this variable is “{"IEEE.std_logic_1164”}”.

---

*Synthesis Variables* 259
vhdlout_wired_and_res_func
Specifies the name of a wired and resolution function. This user-supplied function must be in one of the packages specified by vhdlout_use_packages. If this variable is set to an empty string, a default wired and resolution function is written out, if needed.

Default value for this variable is “”.

vhdlout_wired_or_res_func
Specifies the name of a wired or resolution function. This user-supplied function must be in one of the packages specified by vhdlout_use_packages. If this variable is set to an empty string, a default wired or resolution function is written out, if needed.

Default value for this variable is “”.

vhdlout_write_architecture
When true (the default), write -format vhdl writes architecture declarations. When false, no architecture declarations are written.

Default value for this variable is “true”.

vhdlout_write_components
When true (the default), write -format vhdl writes out component declarations for cells mapped to a technology library. When false, no component declarations are written. Component declarations are required by VHDL format; if you set this variable to false, ensure that vhdlout_use_packages includes a package containing the necessary component declarations.

Default value for this variable is “true”.
**vhdlout_write_entity**
When true (the default), write -format vhdl writes entity declarations. When false, no entity declarations are written.

Default value for this variable is “true”.

**vhdlout_write_top_configuration**
When true, write -format vhdl writes a configuration statement if necessary, as when ports on the top-level design are written as vectors instead of as user's types. The configuration statement’s port map contains calls to type conversion functions. (For more information, see the man pages for vhdlout_single_bit and vhdlout_preserve_hierarchical_types.)

Default value for this variable is “false”.

**vhdlout_zero_name**
Determines the literal name for constant bit value ‘0’ in a design written to VHDL. This is useful when your design methodology is based on a more general logic value than BIT. Used with vhdlout_bit_type. Default is '0'.

Default value for this variable is “0”.

**view_analyze_file_suffix**
This variable is a list of file extensions that specify the files shown in the File/Analyze dialog. The default value is {.vhd, .v, .vhdl}.

Default value for this variable is “{.vhd,.v, .vhdl}”.
view_arch_types
List of host machine architectures that can be used for background jobs from the Design Analyzer viewer. This variable is used to set the contents of the architecture option menu.

Default value for this variable is “{apollo, dcmips, hp300, mips, prism, sgimips, sonymips, sun3, sun4}”.

view_background
Specifies the background color of the Design Analyzer viewer. The valid settings are black (default) and white.

Default value for this variable is “black”.

view_cache_images
When true (the default value), specifies that bitmaps are to be cached for fast schematic drawing.

Default value for this variable is "true".

view_command_log_file
When set, all text written to the Design Analyzer Command Window is written to the specified file.

Default value for this variable is “".

view_command_win_max_lines.
This variable contains the maximum number of lines that are to be saved in the Design Analyzer command window. When more than this number of lines of output are added to the command window, the older lines at the top of the list are removed.

Default value for this variable is “1000".
view_dialogs_modal
When true, the question and error dialogs in Design Analyzer require a confirmation before you can continue to enter commands.
Default value for this variable is “true”.

view_disable_cursor_warping
When false, the cursor is automatically warped (or moved) to dialogs when they are posted. Default is true.
Default value for this variable is “true”.

view_disable_error_windows
When true, the error windows are not posted when errors occur. Default is false.
Default value for this variable is “false”.

view_disable_output
When true, disables output to the Design Analyzer command window. This is useful when running the Design Analyzer over slow networks, such as phone lines. The default value is false.
Default value for this variable is “false”.

view_error_window_count
Maximum number of errors that Design Analyzer reports for a command. If more than the specified number of errors occurs, you are informed that additional errors can be seen in the command window. The error window is suppressed until the end of the command.
Default value for this variable is “6”.

---
Synthesis Variables  263
view_execute_script_suffix
Used by the Execute Script option of the Setup menu and displays only files with these suffixes from directories you select in the option window.

view_info_search_cmd
If this variable is set, the optional menu item On-Line Information allows you to invoke the online information viewer. The value of this variable should be set to the UNIX pathname to the online information viewer.

Default value for this variable is "".

view_log_file
Specifies the file where events that occur in the viewer are stored. This file is useful for error reporting and can be executed with the Execute Script option of the Setup menu, or inserted with the include command in the Design Analyzer.

Default value for this variable is "".

view_on_line_doc_cmd
If this variable is set, the optional menu item On-Line Documentation allows you to invoke the online documentation viewer. The value of this variable should be set to the UNIX command to invoke the online documentation view.

Default value for this variable is "".

view_read_file_suffix
Used by the Read option of the Design Analyzer File menu and displays only files with these suffixes from directories you select in the option window.

Default value for this variable is "{db, sdb, edif, eqn, fnc, lsi, mif, NET, pla, st, tdl, v, vhd, vhdl}".
view_script_submenu_items
Allows users to add to the Setup pulldown menu items for invoking 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, and the second member is the string that gets sent to the dc_shell command line for execution. Any legal dc_shell command sequence can be used.

 Default value for this variable is “{}”.

view_tools_menu_items
A .synopsys_dc.setup file variable that permits partial configuration of the Tools pulldown menu to add a new menu item for invoking user scripts. Contains a list of strings grouped into pairs where the first member of the pair is the text that will appear in the submenu, and the second member is the string that is sent to the dc_shell command line for execution. Any legal dc_shell command sequence can be used.

 Default value for this variable is “{}”.

view_use_small_cursor
Is true if the X-display only supports 16-bit (small) cursors.

view_use_x_routines
When false, internal arc-drawing routines (instead of X routines) are used. If there is a math co-processor chip on the same machine that the X server is on, X arc-drawing routines are faster. Otherwise, internal arc-drawing routines are faster. Default is true.

 Default value for this variable is “true”.

view_write_file_suffix

Used by the Save As option of the File menu and displays only files with these suffixes from directories you select in the option window.

Default value for this variable is “{db, sdb, edif, eqn, fnc, lsi, mif, NET, pla, st, tdl, v, vhd, vhdl}”.

write_name_mapping_nowarn_libraries

Specifies a list of libraries for which no warning messages are to be issued by write -f edif -names_file if the libraries are not found. The default is to issue warning messages for all libraries not found. For example, if the library foo does not exist and write_name_mapping_nowarn_libraries is set to its default value, warning messages are issued saying that the library foo has not been found. However, if write_name_mapping_nowarn_libraries = {foo}, then the warning messages are suppressed. These warning (error) messages include all components in that library and all pins in those components.

Default value for this variable is “{"}”.

write_name_nets_same_as_ports

When true, nets that are connected to ports are given the same names as those ports in the descriptions of designs written in the EDIF, LSI, or TDL format. (Other nets might be renamed to avoid creating shorts.)

Default value for this variable is “false”.

write_test_formats

Specifies the test vector formats recognized and created by the write_test command.

Default value for this variable is “{"synopsys" "tds" "wgl" "verilog" "vhdl"}”.
write_test_include_scan_cell_info
When true (the default value), write_test includes scan-chain/cell/inversion information in the vector files for vector formats that support this information.
Default value for this variable is “true”.

write_test_input_dont_care_value
Controls the logic value output by the write_test command when you have an input with a dont_care condition.
Default value for this variable is “X”.

write_test_max_cycles
The write_test_max_cycles variable allows the user to control the automatic partitioning of long test sets across multiple files by specifying the maximum number of tester cycles to be contained in any one vector file. One tester cycle corresponds to either shifting scan data one bit in the scan chains, or one cycle of parallel (non-shift) operation of the device under test.
Default value for this variable is “0”.

**write_test_max_scan_patterns**

The `write_test_max_scan_patterns` variable allows the user to control the automatic partitioning of long test sets across multiple files by specifying the maximum number of scan-test patterns to be contained in any one vector file. The application of a scan-test pattern encompasses the serial loading and unloading of the scan-chains and, hence, consumes many tester cycles. Typically, this correlates directly to the number of scan-test patterns generated by `create_test_patterns`; however, for designs with multiple system clocks, it may be necessary to repeat the application of each scan-test pattern a number of times.

Default value for this variable is “0”.

**write_test_pattern_set_naming_style**

The `write_test_pattern_set_naming_style` specifies how patterns sets are named when long test sets are partitioned across multiple files. The pattern set name is intended as a documentation aid and is placed inside a comment header in the vector files.

Default value for this variable is “TC_Syn_%d”.

**write_test_round_timing_values**

When true (the default), `write_test` rounds all timing values (for example, input delay, output strobe time, bidir delay, clock period, and clock edge times) to the nearest integer. If you do not want the timing values to be rounded to the nearest integer, set the value of this variable to false.

Default value for this variable is “true”.

---

268 Synthesis Variables
**write_test_scan_check_file_naming_style**

The `write_test_scan_check_file_naming_style` specifies how to name the file containing the vectors which test the scan chain logic.

Default value for this variable is “%s_schk.%s”.

**write_test_vector_file_naming_style**

The `write_test_vector_file_naming_style` specifies how scan vector file names are derived, especially when long test sets must be split across multiple files.

Default value for this variable is “%s_%d.%s”.

**x11_set_cursor_background**

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.

Default value for this variable is “blue”.

**x11_set_cursor_foreground**

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.

Default value for this variable is “red”.

**x11_set_cursor_number**

Specifies the cursor from the standard X cursor font used by the Design Analyzer menus and viewer.

Default value for this variable is “2”.

---

**Synthesis Variables** 269
xnfin_dff_clock_enable_pin_name
This variable tells the read -format xnf command to assume that the clock enable pin name on the DFF (D Flip Flop) component has the given string name. This is used in supporting different versions of the Xilinx library, which can have differing pin names for these sequential devices.

Default value for this variable is “CE”.

xnfin_dff_clock_pin_name
This variable tells the read -format xnf command to assume that the clock pin name on the DFF (D Flip Flop) component has the given string name. This is used in supporting different versions of the Xilinx library, which can have differing pin names for these sequential devices.

Default value for this variable is “C”.

xnfin_dff_data_pin_name
This variable tells the read -format xnf command to assume that the data pin name on the DFF (D Flip Flop) component has the given string name. This is used in supporting different versions of the Xilinx library, which can have differing pin names for these sequential devices.

Default value for this variable is “D”.

xnfin_dff_q_pin_name
This variable tells the read -format xnf command to assume that the Q pin name on the DFF (D Flip Flop) component has the given string name. This is used in supporting different versions of the Xilinx library, which can have differing pin names for these sequential devices.

Default value for this variable is “Q”.

270 Synthesis Variables
**xnfin_dff_reset_pin_name**
This variable tells the read -format xnf command to assume that the reset pin name on the DFF (D Flip Flop) component has the given string name. This is used in supporting different versions of the Xilinx library, which can have differing pin names for these sequential devices.

Default value for this variable is “RD”.

**xnfin_dff_set_pin_name**
This variable tells the read -format xnf command to assume that the set pin name on DFF (D Flip Flop) component has the given string name. This is used in supporting different versions of the Xilinx library, which can have differing pin names for these sequential devices.

Default value for this variable is “SD”.

**xnfin_family**
This variable tells the read -format xnf command to assume that the file being read is a design of the specified Xilinx family. Currently, the XNF reader only supports the 4000 family of Xilinx parts, so this is the only valid value.

Default value for this variable is “4000”.

**xnfin_ignore_pins**
This variable tells the read -format xnf command to leave the given Xilinx component pin names unconnected. This is necessary because post-layout Xilinx designs may have global set and reset pins specified on components. Since these pins do not appear on the symbols for the Xilinx devices, it is necessary to specify that the XNF reader unconnect these pins whenever they are encountered in a netlist.

Default value for this variable is “GTS”.
**xnfout_clock_attribute_style**

This variable controls how the write -format xnf command writes XNF netlist timing constraints. Depending on the value of this variable, the CLOCK_TO_SETUP attributes will be generated with a different style.

Default value for this variable is “CLK_ONLY”.

**xnfout_constraints_per_endpoint**

When set, the write -format xnf command will write out this many timing constraints per endpoint in the design. The purpose is to control the complexity and size of the XNF file which is created. The write -format xnf command writes timing constraints into the XNF netlist format. For each sequential endpoint in the design a set of timing constraints of different types are written. This variable specifies the number of constraints written for each endpoint in the design for a constraint type. An endpoint is either an output port or an input pin to a sequential element. For example, let xnfout_constraints = 5, and a design has endpoints with more than 5 paths of a certain type for an endpoint. Then only the first 4 paths are written as a specific timing constraint. After that, a default timing constraint will be written if the xnfout_default_timing_constraints is set to TRUE. Otherwise, an additional non-default time constraint will be written. A large number will give the most accurate description of constraints, but may require additional memory and CPU time for the place and route program to read and analyze the constraints. A small number will simplify the constraints and could potentially incorrectly specify the design, because default constraints are worst case of all remaining paths. By default, 50 paths per endpoint are
constrained, to achieve an accurate description of the design. Note: If this variable is set to 0 then no timing constraints are written. Setting to zero disables the timing constraints feature. PP Note: Timing constraints will only be written for endpoints which have a required time constraint. If no constraints are written, then check to see that either max_delay or clock_period constraints have been set for the design. For a list of all io variables and their current values, use the list -variables io command.

Default value for this variable is “50”.

**xnfout_default_time_constraints**

When this variable is set to true, the write -format xnf command will write out default time constraints for the paths in the design that are not covered by a specific path timing constraint. When this variable is set to false, then only path specific timing constraint will be written, and no default timing will appear in the XNF format.

Default value for this variable is “true”.

**xnfout_library_version**

When set, the write -format xnf command will write out the LIBVER= attribute with this string onto symbol records.

Default value for this variable is “”.

**xterm_executable**

Specifies the path to an xterm program spawned to run Synopsys analysis tools (for example, RTL Analyzer or BCView). The default is xterm.

Default value for this variable is “xterm”.