Formality®
Quick Reference

Version X-2005.12, December 2005

Comments?
Send comments on the documentation by going to http://solvnet.synopsys.com, then clicking "Enter a Call to the Support Center."

SYNOPSYS®

Copyright Notice and Proprietary Information

Right to Copy Documentation

The license agreement with Synopsys permits licensee to make copies of the documentation for its internal use only. Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any. Licensee must assign sequential numbers to all copies. These copies shall contain the following legend on the cover page:

> "This document is duplicated with the permission of Synopsys, Inc., for the exclusive use of _____ and its employees. This is copy number _____."

Destination Control Statement

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

Disclaimer

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

Printed in the U.S.A.

Document Order Number: 33036-000 A1
Formality Quick Reference, version X-2005.12

Getting Help

Formality provides various forms of online help. The help command provides you with quick help for one or more commands or procedures. Unlike Synopsys Design Compiler, which allows you to use the man and help commands interchangeably for viewing man pages, man in Formality displays the man page, and help provides online assistance.

You can use a wildcard pattern as the argument for the help command. The wildcard characters are

* Matches *n* characters.

? Matches exactly one character.

Accessing Brief Help

Use this command to list all commands by function group:

```
fm_shell> help
```

Use this command to display all commands that end with the word clock:

```
fm_shell> help *clock
```

Use this command to get syntax help for one or more commands:

```
fm_shell> help -verbose command_name_pattern
```

Use this command to get syntax help for a specific command:

```
fm_shell> command_name -help
```

Printing Variables

Invoke these commands from within the Formality tool.

Use this command to list the value of environment variables:

```
fm_shell> printenv [variable_name]
```

Use this command to list the values of one or more variables:

```
fm_shell> printvar [pattern]
```

Man Page Viewing Instructions

The following sections describe how to set up your environment and the syntax to use to view man pages.

Setting up the UNIX environment

Edit your .cshrc file to contain these lines:

```
setenv FM_MAN_DIR formality_root/doc/fm/man
setenv MANPATH ${MANPATH}:${FM_MAN_DIR}
```

FM_MAN_DIR is a variable that contains the path to the man page directories, and *formality_root* represents the specific path to the Formality software directory at your site.

Viewing Man Pages from UNIX

Command

```
% man command_name
```

Variable

```
% man variable_name
```

Error, warning, or information message

```
% man message_id
```

Viewing Man Pages from fm_shell

Command

```
fm_shell> man command_name
```

Variable

```
fm_shell> man variable_name
```

Error, warning, or information message

```
fm_shell> man message_id
```

Viewing Man Pages from the GUI

You can use the man page viewer in the GUI to view command, variable, and error messages. Clicking the man page view button from the main menu bar in the GUI brings up the Man Page Viewer, where you can click a link to the commands, variables, or messages man pages. From there, you click the specific man page you want.

User Commands and Variables

Invoke user commands and variables from a UNIX shell.

fm_shell

Runs the Formality shell.

```
fm_shell
[ -root_path path_name ]
[ -32bit] [ -64bit ]
[ -file file_name ]
[ -no_init ]
[ -session session_file_name ]
[ -version ]
[ -gui ]
```

formality

Runs the Formality GUI.

```
formality
[ -root_path path_name ]
[ -32bit] [ -64bit ]
[ -root_path path_name ]
[ -session session_file_name ]
[ -file file_name ]
[ -no_init ]
[ -version ]
```

You can also invoke the GUI from within the `fm_shell` by specifying the `start_gui` command.

FM_WAIT_LICENSE

Adds a request for a queue on a license server. Default: "0”

FM_WAIT_LICENSE { 1 | 0 }

Formality Commands

Invoke these commands from within the Formality tool. In addition to the commands listed here, there are also many Tcl and UNIX commands available from the Formality shell.

add_distributed_processors

Sets up a distributed processing environment and allows distributed equivalence checking verification.

```
add_distributed_processors
machine machine ...
[ -lsf ]
-nservers
-options
```

alias

Creates a pseudo-command that expands to one or more words, or lists current alias definitions.

```
alias
[name]
[def]
```

cputime

Returns the CPU time used by the Formality shell.

```
cputime
```

create_constraint_type

Creates a named user-defined (arbitrary) constraint type.

```
create_constraint_type
type_name designID
```

create_container

Creates a new container.

```
create_container
containerID
```

create_cutpoint_blackbox

This command is obsolete beginning with the 2004.06 release. Use set_cutpoint in its place.

current_container

Establishes or reports the current container.

```
current_container
[ containerID ]
```

current_design

Establishes or reports the current design.

```
current_design
[ designID ]
```

debug_library_cell

Debugs the specified library cell in library_verification mode.

```
debug_library_cell
[ cell_name ]
```

diagnose

Runs diagnosis on the most recent verification.

```
diagnose
[ -all | failing point list]
[ -effort_level low | medium | high ]
[ -pattern_limit limit ]
```

echo

Echos arguments to standard output.

```
echo
[-n]
[Iargument...]
```

elaborate_library_cells

Resolves cell references before verifying the library.

```
elaborate_library_cells
```

error_info

Prints extended information on errors from last command.

```
error_info
```

exit

Exits the Formality shell.

```
exit
```

find_cells

Lists cells in the current design.

```
find_cells
[ -of_object objectID ]
[ -library_cells | -nolibrary_cells ]
[ -type ID_type ]
[ cellID_list ]
```

find_designs

Returns a list of designs.

```
find_designs
[ object_list ]
[ -passing ][ -failing ][ -aborted ]
[ -not_verified ]
```

find_nets

Returns a list of nets.

```
find_nets
[ object_list ]
[ -hierarchy ]
find_nets
[ -of_object objectID ]
[ -type ID_type ]
```

find_pins

Returns a list of pins.

```
find_pins
[ -of_object objectID ]
[ -in ] [ -out ] [ -inout ]
[ -hierarchy ]
[ -type ID_type ]

find_pins
[ object_list ]
[ -in ] [ -out ] [ -inout ]
```

find_ports

Returns a list of ports.

```
find_ports
[ -of_object objectID ]
[ -in ] [ -out ] [ -inout ]
[ -type ID_type ]

find_ports
[ object_list ]
[ -in ] [ -out ] [ -inout ]
```

find_references

Returns a list of designs instantiated within the specified designs.

```
find_references
[ -of_object objectID ]

find_references
[ design_list ]
[ -black_box ]
[ -hierarchy ]
```

get_unix_variable

Returns the value of a UNIX environment variable.

```
get_unix_variable
variable_name
```

group

Groups any number of cells or instances in the current design into a new design, creating a new level of hierarchy.

```
group
cell_list -design_name design_name
[ -cell_name cell_name]
```

guide

Puts Formality into guide mode, supporting guidance for all the SVF features.

```
guide
```

guide_architecture_db

Guides Formality on the association of a db file with an architectural implementation.

```
guide_architeture_db
[ -file file_name ] [ libraries ]
```

guide_architecture_netlist

Guides Formality on the association of a netlist file with an architectural implementation.

```
guide_architeture_netlist
[ -file file_name ] [ libraries ]
```

guide_change_names

Guides Formality on changing the name of design objects.

```
guide_change_names
-design designName
[ -instance instanceName ] [ changeBlock]
```

guide_datapath

Guides Formality in identifying a datapath subdesign.

```
guide_datapath
-design designName
[ -instance instanceName ] [ changeBlock]
```

guide_fsm_reencoding

Guides Formality on the changes in the encoding of the finite-state machine.

```
guide_fsm_reencoding
-design design_name
-previous_state_vector prevList
-current_state_vector currList
-state_reencoding stateList
```

guide_group

Guides Formality on the hierarchy created around design objects.

```
guide_group
-design design_name
[ -instance instanceName
[ -cells cellList ]
-new_design newDesignName
-new_instance newInstanceName
[ groupBlock ]
```

guide_multiplier

Guides Formality in identifying a subdesign as a multiplier with a specific architecture.

```
guide_multiplier
-design design_name
[ -instance instanceName
-arch arch
-body fileName
```

guide_reg_constant

Guides Formality in identifying a constant register.

```
guide_reg_constant
[ -design designName ]
instanceName
constantVal
```

guide_reg_duplication

Guides Formality in identifying duplicate registers.

```
guide_reg_duplication
[ -design designName ]
-from fromReg
-to toList
```

guide_reg_merging

Guides Formality in identifying merged registers.

```
guide_reg_merging
[ -design designName ]
-from fromReg
-to toList
```

guide_ungroup

Guides Formality to remove hierarchy from design objects.

```
guide_ungroup
-design designName
[ -instance instanceName ]
[ -cells cellList ]
[ -all ]
[ -small ]
[ -flatten ]
[ -start_level ]
[ ungroupBlock
```

guide_uniquify

Guides Formality in creating unique instances for design objects.

```
guide_uniquify
-design designName
[ uniquifyBlock ]
```

guide_ununiquify

Integrates uniquified instances back into their original designs.

```
guide_ununiquify
-design designName
[ ununiquifyBlock ]
```

help

Returns information about Formality shell commands.

```
help
[ command ]
```

history

Displays or modifies the commands recorded in the history list.

```
history
[-h]
[-r]
[args...]
```

library_verification

Transfers Formality from one of the design verification modes to cell library verification mode.

```
library_verification
mode
```

list_key_bindings

Displays all the key bindings and edit mode of the current shell session.

```
list_key_bindings
[ -nosplit ]
```

list_libraries

Lists technology libraries currently loaded.

```
list_libraries
```

man

Displays man pages for Formality shell commands.

```
man
[ command ]
```

match

Causes Formality to perform compare point matching.

```
match
[ -datapatch ]
[ -hierarchy ]
```

memory

Reports memory used by the Formality shell.

```
memory
```

printenv

Prints the value of environment variables.

```
printenv
variable_name
```

printvar

Prints the values of one or more variables.

```
printvar
pattern
```

proc_args

Displays the formal parameters of a procedure.

```
proc_args
proc_name
```

proc_body

Displays the body of a procedure.

```
proc_body
proc_name
```

quit

Exits the Formality shell.

```
quit
```

read_container

Reads a previously written container into the Formality environment.

```
read_container
[ -r | -i | -container containerID ]
[ -replace ]
file
```

read_db

Reads Synopsys *.db designs or technology libraries.

```
read_db
[ -r | -i | -container containerID ]
[ -libname library_name ]
[ -technology_library ]
[ -merge ]
[ -replace_black_box ]
file_list
```

read_ddc

Reads designs from a .ddc database.

```
read_ddc
[ -r | -i | -container containerID ]
[ -libname library_name ]
[ -technology_library ] file_list
```

read_edif

Reads one or more EDIF files into the Formality environment.

```
read_edif
[ -r | -i | -container containerID ]
[ -libname library_name ]
[ -work_library library_name ]
[ -technology_library ]
file_list
```

read_fsm_states

Reads finite-state machine (FSM) states.

```
read_fsm_states
file
[ designID ]
```

read_mdb

Reads designs from a Milkyway database.

```
read_mdb
[ -r | -i | -container containerID ]
[ -libname library_name ]
[ -technology_library ]
-cell_name cell_name
[ -design design_name ]
mdb_path_name
```

read_simulation_library

Reads a Verilog simulation library into the Formality environment.

```
read_simulation_library
[ -r | -i | -container containerID ]
[ -libname library_name ]
[ -work_library library_name ]
[ -skip_unused_udps ]
[ -write_verilog ]
[ -save_binary ]
[ -merge ]
[ -replace_black_box ]
[ -halt_on_error ]
file_list
```

read_verilog

Reads one or more Verilog files into the Formality environment.

```
read_verilog
[ -r | -i | -container containerID ]
[ -libname library_name ]
[ -work_library library_name ]
[ -netlist ]
[ -technology_library ]
[ -f vcs_option_file ]
[ -vcs "VCS options" ]
[ -define ]
[ -01 | -95 ]
file_list
```

read_vhdl

Reads one or more VHDL files into the Formality environment.

```
read_vhdl
[ -r | -i | -container containerID ]
[ -libname library_name ]
[ -work_library library_name ]
[ -technology_library ]
[ -87 | -93 ]
file_list
```

redirect

Redirects the output of a command to a file.

```
redirect
[ -append ]
file_name
{ command }
string file_name
string command
```

remove_black_box

Removes user-defined, design-specific black boxes.

```
remove_black_box
[ design_list ]
-all
```

remove_compare_rules

Removes all user-defined compare rules.

```
remove_compare_rules
[ designID ]
```

remove_constant

Removes one or all user-defined constants.

```
remove_constant -all
remove_constant
[ -type ID_type ] instance_path...
```

remove_constraint

Removes external constraints from the control points of a design.

```
remove_constraint
constraint_name
```

remove_constraint_type

Removes named user-defined external constraint types created using the create_constraint_type command.

```
remove_constraint_type
type_name
```

remove_container

Removes one or more containers from the Formality environment.

```
remove_container
container_list
remove_container -all
```

remove_cutpoint

Removes cutpoints previously specified with the set_cutpoint command.

```
remove_cutpoint object_list
[ -type objectID_type ]

remove_cutpoint -all
```

remove_design

Removes designs from the Formality environment and replaces them with a black box.

```
remove_design
[ -hierarchy ]
[ -shared_lib]
design_list
```

remove_design_library

Removes one or more design libraries from the Formality environment.

```
remove_design_library
library_list

remove_design_library -all
```

remove_distributed_processors

Removes servers for the distributed processing, which were added using add_distributed_processors.

```
remove_distributed_processors
machine machine ... | -all
```

remove_dont_verify_point

Removes a list of user-defined, don't-verify points.

```
remove_dont_verify_point
[ -type ID_type ]
[ objectID_list ]

remove_dont_verify_point -all
```

remove_equivalence

Removes one or all user-defined equivalences.

```
remove_equivalence
[ -type ID_type ]
objectID_1
objectID_2

remove_equivalence -all
```

remove_inv_push

Removes user-defined inversion push.

```
remove_inv_push
[ -shared_lib ]
objectID_list

remove_inv_push -all
```

remove_library

Removes one or more technology libraries from the Formality environment.

```
remove_library
library_list

remove_library -all
```

remove_object

Removes pins, ports, or unlinked cells.

```
remove_object
objectID
[ -shared_lib]
[ -type ID_type ]
```

remove_parameters

Removes user-defined, design-specific parameters.

```
remove_parameters
parameter_list
[ designID_list ... ]
-all_designs
```

remove_resisitive_drivers

Removes pullups, pulldowns, bus-holders, etc.

```
remove_resisitive_drivers
```

remove_user_match

Removes user_specified matches.

```
remove_user_match
[ -type ID_type ]
[ instance_path ]

remove_user_match
[ -all ]
```

rename_object

Renames one or more design objects.

```
rename_object
objectID
-file filename
[ new_name ]
[ -type ID_type ]
[ -shared_lib ]
[ -reverse ]
[ -container containerID ]
```

report_aborted_points

Produces information about compare points not verified.

```
report_aborted_points
[ -compare_rule ]
[ -substring string ]
[ -point_type point_type ]
[ -loop ] [ -hard ]
-status [ inverted | noninverted ]
```

report_architecture

Displays the architecture used to implement the specified instance as well as what caused the selection of that instance.

```
report_architecture
[ -set_architecture |
[ -hdlin_multiplier_architecture |
-fm_pragma | -all | instance_path ]
```

report_black_boxes

Generates information about black boxes.

```
report_black_boxes
[ design_list | -r | -i | -con containerID ]
[ -all | -unresolved | -interface_only |
-empty | -set_black_box ]
```

report_cell_list

Reports library cells depending on the option specified. This command is available only in the library verification mode.

```
report_cell_list
[ -r | -i ]
[ -matched ] [ -unmatched ]
[ -verify ]
[ -passing ] [ -failing ] [ -aborting ]
[ -filter wildcard_pattern ]
```

report_compare_points

This command is obsolete. Use the report_user_matches and report_dont_verify_points commands instead.

report_compare_rules

Produces information about user-defined compare rules.

```
report_compare_rules
[ designID ]
```

report_constants

Produces information about user-defined constants.

```
report_constants
[ objectID ...]
```

report_constraint

Provides information about constraints.

```
report_constraint
[constraint_name]
[ -long]
```

report_constraint_type

Provides information about constraint types.

```
report_constraint_type
[type_name]
[ -long]
```

report_containers

Produces a list of containers.

```
report_containers
```

report_cutpoints

Reports all the cutpoints you have specified.

```
report_cutpoints
```

report_design_libraries

Produces information about design libraries.

```
report_design_libraries
[ item_list ]
```

report_designs

Produces information about designs.

```
report_designs
[ item_list ]
```

report_distributed_processors

Reports all servers currently in the list of distributed servers and identifies the distributed working directory.

```
report_distributed_processors
```

report_dont_verify_points

Produces information about user-defined, don’t verify points.

```
report_dont_verify_points
```

report_environment

Produces information about user-defined parameters that affect the verification and simulation environment. This command will be obsolete in a future release. Use the printvar Tcl command instead of using this command.

```
report_environment
```

report_equivalences

Produces information about user-defined equivalences.

```
report_equivalences
```

report_error_candidates

Produces information about error candidates for the most recent verification.

```
report_error_candidates
[ -p percentage ]
```

report_failing_points

Produces information about compare points that fail verification.

```
report_failing_points
[ -compare_rule ]
[ -substring string ]
[ -point_type point_type ]
[ -matched ] [ -unmatched ]
-status [ inverted | noninverted ]
```

report_fsm

Produces information about state encoding.

```
report_fsm
[ designID | -name FSM_name ]
```

report_guidance

Reports the name, and optionally the contents or disposition, of the previously set SVF.

```
report_guidance
[ -datapath [ -long ] ] [ -to file_name ]
```

report_hdlin_mismatches

Sumarizes the RTL simulation and synthesis mismatches encountered during design linking.

```
report_hdlin_mismatches
[ -summary | -verbose ]
[ -reference ] [-implementation ]
[-container container_name ]
```

report_hierarchy

Produces information about the hierarchy of a design.

```
report_hierarchy
[ designID ]
```

report_inv_push

Produces information about user-defined inversion push.

```
report_inv_push
[ designID ]
```

report_libraries

Produces information about technology libraries.

```
report_libraries
[ -short ]
[ library_list ]
```

report_loops

Reports loops or portions of loops in either the reference or implementation designs from the last verification run.

```
report_loops
[ -ref | -impl ]
[-limit N ]
[ -unfold ]
```

report_matched_points

Produces information about all matchable design objects that are matched.

```
report_matched_points
[ -compare_rule ]
[ -datapath ]
[ -substring string ]
[ -point_type point_type ]
[ -method matching_method ]
[ -status status ]
[ -except_status status ]
[ -last ]
[[ -type ID_type ] objectID ]
```

report_parameters

Produces information about user-defined parameters.

```
report_parameters
[ design_list ]
```

report_passing_points

Produces information about compare points that passed the most recent verification.

```
report_passing_points
[ -compare_rule ]
[ -substring string ]
[ -point_type point_type ]
[ -status status ]
```

report_power_gating

Reports the results of the power-gating comparison.

```
report_power_gating
```

report_status

Reports the current status of verification in either normal mode for in library verification mode. The options are only available in library verification mode.

```
report_status
[-pass ]
[ -fail ]
[ -abort ]
```

report_svf

This command is obsolete. Instead, use the report_guidance command.

report_truth_table

Generates and prints a truth table for a given signal.

```
report_truth_table
[ -display_fanin ]
[ -fanin {list of signals} ]
[ -constraints {signal=[0/1]}+ ]
[ -nb_lines int ]
[ -max_line int ]
[ -max_fanin int ]
```

report_unmatched_points

Reports points that have not been matched.

```
report_unmatched_points
[ -compare_rule ]
[ -datapath ]
[ -substring string ]
[ -point_type point_type ]
[ -status status ]
[ -except_status status ]
[[ -type ID_type ] objectID... ]
```

report_unverified_points

Reports points that were not verified during compare point matching.

```
report_unverified_points
[ -compare_rule ]
[ -substring string ]
[ -point_type point_type]
[ -cause cause ]
[ -status status ]
[ -inputs input_type ]
```

report_user_matches

Produces information about user-defined matched points.

```
report_user_matches
[ -inverted ]
[ -noninverted ]
[ -unknown ]
```

report_vhdl

Produces a list of VHDL configurations, entities, architectures, and associated generics and their default values.

```
report_vhdl
[ -switches ]
[ -name ]
[ -configuration ]
[ -entity ]
[ -package ]
```

restore_session

Restores a previously saved Formality session.

```
restore_session
file
```

save_session

Saves the current Formality session.

```
save_session
[ -replace ]
file
```

select_cell_list

Selects library cells depending on the option specified.

```
select_cell_list
[ cellNames/wildcards ]
[ -filename ]
[ -add cellNames/wildcards ]
[ -remove cellNames/wildcards ]
[ -clear ]
```

set_architecture

Instructs Formality to replace the multiplier architecture of the specified instance with the architecture defined in the command.

```
set_architecture
[ InstanceName ]
[ csa | nbw | wall | dw_foundation ]
```

set_black_box

Establishes black boxes for specified designs.

```
set_black_box
[ designID_list ]
```

set_compare_rule

Adds a name matching rule that Formality applies to a design before creating compare points.

```
set_compare_rule
[ designID ]
-from search_pattern
-to replace_pattern
[ -type ID_type ]
-file filename
```

set_constant

Creates a user-defined constant by setting the logic state of a design object to 0 or 1.

```
set_constant
[ -type ID_type ]
objectID
state
```

set_constraint

Creates an external constraint on a design.

```
set_constraint
type_name
control_point_list
[ designID ]
[ -name constraint_name ]
[ -map mapping_list ]
```

set_cutpoint

Specifies that the given hierarchical pin or net is a hard cutpoint that should be verified independently, and can be used as a free variable for verification of downstream compare points.

```
set_cutpoint
[ -type ID_type ]
objectID
```

set_direction

Defines port or pin directions.

```
set_direction
[ -type ID_type ]
objectID
direction
[ -shared_lib ]
```

set_dont_verify_point

Sets specified compare points to a don't-verify status.

```
set_dont_verify_point
[ -type ID_type ]
[ -directly_undriven_output ]
[ object_1 [ object_2 ]...]
```

set_equivalence

Declares that two nets or ports have equivalent functions.

```
set_equivalence
[ -type ID_type ]
[ -propagate ]
[ -inverted ]
objectID_1
objectID_2
```

set_fsm_encoding

Enumerates FSM state names and their encodings.

```
set_fsm_encoding
encoding_list
[ designID ] -name
```

set_fsm_state_vector

Names state vector flip-flops in an FSM.

```
set_fsm_state_vector
flip-flop_list
[ designID ] -name
```

set_implementation_design

Establishes the implementation design.

```
set_implementation_design
[ designID ]
```

set_inv_push

Adds a sequential object or a top-level port to the list of objects through which Formality transports inverters for verification.

```
set_inv_push
[ -shared_lib ]
objectIDlist
```

set_parameters

Establishes verification parameters for a specific design.

```
set_parameters
[ -flatten ]
[ -resolution function ]
[ -retimed ]
[ designID ]
```

set_power_gating_style

Sets the power_gating_style attribute on designs or HDL blocks, specifying the kind of retention register cells expected.

```
set_power_gating_style
[ -hdl_blocks name ]
-type type
```

set_reference_design

Establishes the reference design.

```
set_reference_design
[ designID ]
```

set_svf

Specifies the name of the SVF (setup verification for Formality).

```
set_svf
[ -append ]
[ -ordered ]
[ -extension name ]
[ filedirnames ]
```

set_top

Resolves cell references and elaborates RTL designs.

```
set_top
[ -vhdl_arch architecture_name ]
[ designID | -auto ]
[ -parameter value ]
```

set_unix_variable

Sets the value of a UNIX environment variable.

```
set_unix_variable
variable_name
new_value
```

set_user_match

Creates a pair of matched points when two comparable design objects are specified.

```
set_user_match
[ -type ID_type ]
objectID_1 objectID_2
[ -inverted ]
[ -noninverted ]
```

set_vsdc

Specifies the location of V-SDC information.

```
set_vsdc
[ -append ]
[ -ordered ]
[ -extension name ]
[ filedirnames ]
```

setup

Causes Formality to revert to setup mode.

```
setup
```

sh

Executes a command in a child process.

```
sh
[ args ]
```

simulate_patterns

Simulates the implementation and reference designs by using previously failing patterns as input vectors.

```
simulate_patterns
[ -no_link ]
file
```

source

Reads a file and evaluates it as a Tcl script.

```
source
[ -echo ]
[ -verbose ]
file
```

start_gui

Invokes the Formality GUI.

```
start_gui
```

stop_gui

Exits the Formality GUI.

```
stop_gui
```

test_compare_rule

Tests a name matching rule on current unmatched points or user-specified names.

```
test_compare_rule
[ designID | -r | -i ]
-from search_pattern
-to replace_pattern
-substring string

test_compare_rule
-name name_list
-from search_pattern
-to replace_pattern
```

translate_instance_pathname

Translates an instance-based path name to a Formality designID or objectID.

```
translate_instance_pathname
[ -type ID_type ] pathname
```

unalias

Removes one or more aliases.

```
unalias
pattern
```

undo_match

Causes Formality to unmatch previously matched design objects.

```
undo_match
[ -all ]
```

ungroup

Removes a level of hierarchy.

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

uniquify

Creates unique design names for multiply instantiated designs in hierarchical designs.

```
uniquify
[ designID ]
```

verify

Causes Formality to prove or disprove functional equivalence given two designs or two comparable design objects.

```
verify
[ designID_1 designID_2 ]
[ [-inverted] |
[ -type ID_type ] objectID_1 objectID_2 ]
[[ -constant0 | constant1 ]
[ -type ID_type ] objectID ]
[ -restart | -incremental ]
[ -level integer ]
```

which

Locates a file and displays its path name.

```
which
filename_list
```

write_container

Saves the information in the current or specified container to a file.

```
write_container
[ -r | -i | -container container_name ]
[ -quiet ]
[ -replace ]
filename
```

write_failing_patterns

Saves failing patterns from the most recent diagnosis.

```
write_failing_patterns
[ -diagnosis_patterns ]
[ -verilog ]
[ -replace ]
filename
```

write_hierarchical_verification_script

Writes a Tcl script to perform hierarchical verification on the current reference and implementation designs.

```
write_hierarchical_verification_script
[ -replace ]
[ -noconstant ]
[ -noequivalence ]
[ -match type ]
[ -save_directory pathname ]
[ -save_file_limit integer ]
[ -save_time_limit integer ]
[ -level integer ]
filename
```

write_library_debug_scripts

Debugs failing cells from library verification mode in the standard Formality design format.

```
write_library_debug_scripts
[ -dir directory_name ]
```

Formality Tcl Variables

Use these Tcl variables to condition the environment from the Formality shell.

architecture_selection_precedence

Informs the arithmetic generator which architecture selection method takes precedence. Default: “ ”

architecture_selection_precedence string

bus_dimension_separator_style

Sets the VHDL/Verilog read option that controls the bus dimension separator style used with multi-dimensional buses. Default: “][”

bus_dimension_separator_style *string*

bus_extraction_style

Sets the VHDL/Verilog read option that controls the bus extraction style. Default: “%s[%d:%d]”

bus_extraction_style *string*

bus_naming_style

Sets the VHDL/Verilog read option that controls the bus naming style. Default: “%s[%d]”

bus_naming_style *string*

bus_range_separator_style

Sets the EDIF/VHDL/Verilog read option that controls the bus range separator style. Default: “:”

bus_range_separator_style *string*

diagnosis_enable_error_isolation

Enables the precise diagnosis engine in Formality, requiring you to manually instruct Formality to run diagnosis. Default: “false”

diagnosis_enable_error_isolation
true | false

diagnosis_pattern_limit

Sets the maximum number of failing patterns that Formality uses during diagnosis. Default: 256

diagnosis_pattern_limit *integer*

distributed_64bit_mode

Determines which type of server executable to use on 64-bit capable machines. Default: “false”

distributed_64bit_mode *string*

distributed_processor_start_timeout

Defines the time allowed for the master to start a Formality server process and receive acknowledgement from the server. Default: “55”

distributed_processor_start_timeout *integer*

distributed_verification_mode

Allows Formality to perform distributed verification when using the verify command. Default: “enable”

distributed_verification_mode *string*

distributed_work_directory

Sets the directory that will be used for the communicaton between the master and the servers. Default: “FM_WORK/server”

distributed_work_directory *string*

dw_foundation_threshold

Helps select the architecture to build for the multiplier architecture based on the value of the dw_foundation stated in the hdlin_multpilier_architecture variable. Default: “52”

DW_foundation_threshold *value*

edifin_blackbox_library_cells

Enables Formality to create black boxes for EDIF library cells that do not contain contents. Default: “false”

edifin_blackbox_library_cells
`true | false`

edifin_ground_cell_name

Specifies the name of the ground cell. Default: “logic_0”

edifin_ground_cell_name *name*

edifin_ground_net_name

Specifies the name of the ground net. Default: “GND”

edifin_ground_net_name *name*

edifin_power_and_ground_representation

Specifies the power and ground representation as either “cell” or “net.” Default: “net”

edifin_power_and_ground_representation
`cell | net`

edifin_power_cell_name

Specifies the name of the power cell. Default: “logic_1”

edifin_power_cell_name *name*

edifin_power_net_name

Specifies the name of the power net. Default: “VDD”

```
edifin_power_net_name name
```

enable_multiplier_generation

Enables Formality to generate multiplier architectures based on user directives. Default: “true”

```
enable_multiplier_generation
true | false
```

enable_power_gating

Activates the pre-verification retention register consistency check. Default: “false”

```
enable_multiplier_generation
true | false
```

gui_report_length_limit

Specifies a GUI report size limit. Default: “30000”

```
gui_report_length_limit integer
```

hdlin_auto_netlist

Enables automatic use of the Verilog netlist reader by the read_verilog command. Default: “true”

```
hdlin_auto_netlist
true | false
```

hdlin_auto_top

Enables automatic detection and linking of the top-level design by the read commands. Default: “false”

```
hdlin_auto_top
true | false
```

hdlin_disable_tetramax_define

Controls whether the Verilog TetraMAX macro is automatically defined for all read_simulation_library commands. Default: “false”

hdlin_disable_tetramax_define
`true | false`

hdlin_do_inout_port_fixup

Controls whether DDX-7 error messages will be generated at link time. Default: “true”

hdlin_do_inout_port_fixup
`true | false`

hdlin_dwroot

Lets Formality know where the SYNOPSYS tree is that contains the DesignWare components instantiated in any of the hierarchies that were read in. Default: “ ”

hdlin_dwroot *value*

hdlin_dyn_array_bnd_check

Controls whether logic is added to check the validity of array indices. Default: “verilog”

hdlin_dyn_array_bnd_check
`none | both | vhdl | verilog`

hdlin_enable_verilog_assert

Controls whether SystemVerilog ASSERT (and related) statements in Verilog 95 and 2001 source files are treated as errors (the default) or ignored. Default: “false”

hdlin_enable_verilog_assert `true | false`

hdlin_error_on_mismatch_message

Works with the hdlin_warn_on_mismatch variable to control the severity of messages alerting you to possible mismatches between simulation and synthesis. Default: “true”

```
hdlin_error_on_mismatch_message
true | false
```

hdlin_ignore_builtin

Controls whether or not to ignore the built_in directive in VHDL files. Default: “false”

```
hdlin_ignore_builtin true | false
```

hdlin_ignore_dc_script

Controls whether or not to ignore the dc_script_begin and dc_script_end directives in Verilog/VHDL files. Default: “false”

```
hdlin_ignore_dc_script true | false
```

hdlin_ignore_full_case

Controls whether or not to ignore the full_case directive in Verilog files. Default: “true”

```
hdlin_ignore_full_case true | false
```

hdlin_ignore_label

Controls whether or not to ignore the label directive in VHDL files. Default: “false”

```
hdlin_ignore_label true | false
```

hdlin_ignore_label_applies_to

Controls whether or not to ignore the label_applies_to directive in Verilog/VHDL files. Default: “true”

```
hdlin_ignore_label_applies_to
true | false
```

hdlin_ignore_map_to_entity

Controls whether or not to ignore the map_to_entity directive in VHDL files. Default: “true”

```
hdlin_ignore_map_to_entity true | false
```

hdlin_ignore_map_to_module

Controls whether or not to ignore the map_to_module directive/attribute in Verilog/VHDL files. Default: “true”

```
hdlin_ignore_map_to_module true | false
```

hdlin_ignore_map_to_operator

Controls whether or not to ignore the map_to_operator directive in Verilog/VHDL files. Default: “true”

```
hdlin_ignore_map_to_operator
true | false
```

hdlin_ignore_parallel_case

Controls whether or not to ignore the parallel_case directive in Verilog files. Default: “true”

```
hdlin_ignore_parallel_case true | false
```

hdlin_ignore_resolution_method

Controls whether or not to ignore the resolution_method directive in VHDL files. Default: “true”

```
hdlin_ignore_resolution_method
true | false
```

hdlin_ignore_synthesis

Controls whether or not to ignore the synthesis_off and synthesis_on directives in VHDL files. Default: “false”

```
hdlin_ignore_synthesis true | false
```

hdlin_ignore_translate

Controls whether or not to ignore the translate_off and translate_on directives in Verilog/VHDL files. Default: “false”

```
hdlin_ignore_translate true | false
```

hdlin_infer_mux

Controls MUX_OP inference for your Verilog/VHDL description. Default: “none”

```
hdlin_infer_mux default | none | all
```

hdlin_interface_only

Loads the specified designs as black boxes (pin names and directions only). Default: “ ”

```
hdlin_interface_only design(s)
```

hdlin_library_directory

Designates all designs contained within the specified directories as TECHNOLOGY cells. Default: “ ”

```
hdlin_library_directory string
```

hdlin_library_enhanced_analysis

Controls the analysis of Verilog switch primitaves. Default: “false”

```
hdlin_library_enhanced_analysis
true | false
```

hdlin_library_file

Designates all designs contained within the specified files as TECHNOLOGY cells. Default: “ ”

```
hdlin_library_file string
```

hdlin_multiplier_architecture

Elaborates all multipliers using the specified multiplier architecture in the file specified when one of the read commands is run.
Default: “none”

```
hdlin_multiplier_architecture
none | csa | nbw | wall | dw_foundation
```

hdlin_normalize_blackbox_busses

Controls how Formality names black box busses during the link operation. Default: “false”

```
hdlin_normalize_blackbox_busses
true | false
```

hdlin_synroot

Points to the top-level directory of the Design Compiler verion you used to generate MDB or DDC databases. Default: “”

```
hdlin_synroot name
```

hdlin_unresolved_modules

Specifies how to control black box creation for unresolved design references. Default: “error”

```
hdlin_unresolved_modules
black_box | error
```

hdlin_verilog_95

Controls the default Verilog language interpretation of the Formality RTL reader. Default: “false”

```
hdlin_verilog_95 true | false
```

hdlin_verilog_wired_net_interpretation

Specifies whether non-wire nets are resolved using a simulation or a synthesis interpretation. Default: “simulation”

```
hdlin_verilog_95 simulation | synthesis
```

hdlin_vhdl_87

Controls the default VHDL language interpretation of the Formality RTL reader. Default: “false”

```
hdlin_vhdl_87 true | false
```

hdlin_vhdl_auto_file_order

Enables read_vhdl to automatically order the specified files. Default: “true”

```
hdlin_vhdl_auto_file_order
true | false
```

hdlin_vhdl_forgen_inst_naming

Specifies the scheme that should be used to name component instances within VHDL for -generate statements. Default: “mode0”

```
hdlin_vhdl_forgen_inst_naming
mode1 | mode2 | mode3
```

hdlin_vhdl_fsm_encoding

Specifies the fsm encoding for your VHDL description. Default: “binary”

```
hdlin_vhdl_fsm_encoding
binary | 1hot
```

hdlin_vhdl_integer_range_constraint

Controls whether Formality adds logic to constrain the value of integer ports and registers to match the integer rang constraint specified in the VHDL. Default: “false”

```
hdlin_vhdl_integer_range_constraint
true | false
```

hdlin_vhdl_others_covers_extra_states

Controls if Formality uses others clauses to determine how unreachable states are handled. Default: “true”

```
hdlin_vhdl_others_covers_extra_states
true | false
```

hdlin_vhdl_presto_naming

Controls the generation of operator names inferred from VHDL. Default: “false”

```
hdlin_vhdl_presto_naming true| false
```

hdlin_vhdl_presto_shift_div

Controls the implementation of a divide by a constant power of two. Default: “false”

```
hdlin_vhdl_presto_shift_div true| false
```

hdlin_vhdl_strict_libs

Controls whether a strict mapping of VHDL libraries will be used during synthesis. Default: “false”

```
hdlin_vhdl_strict_libs true| false
```

hdlin_vhdl_use_87_concat

Controls whether the IEEE Std 1076-1987 style concantenations are used in the IEEE Std 1076-1993 VHDL code. Default: “false”

```
hdlin_vhdl_use_87_concat true| false
```

hdlin_warn_on_mismatch_message

Works with the hdlin_error_on_mismatch variable to control the severity of messages alerting you to possible mismatches between simulation and synthesis. Default: “ ”

```
hdlin_warn_on_mismatch_message
message_id_list
```

impl

Set to indicate the current implementation design. Default: “ ”

```
impl
```

message_level_mode

Sets the message severity threshold that Formality uses during verification. Default: “info”

```
message_level_mode
error | warning | info
```

message_x_source_reporting

Turns on reporting for all nets that are possible X sources in the designs being verified. Default: “false”

```
message_x_source_reporting
true | false
```

mw_logic0_net

Specifies the name of the Milkyway ground net. Default: “VSS”

```
mw_logic0_net
```

mw_logic1_net

Specifies the name of the Milkyway power net. Default: “VDD”

```
mw_logic1_net
```

name_match

Controls whether compare point matching uses object names, or relies solely on function and topology to match compare points. Default: “all”

```
name_match all | none | port | cell
```

name_match_allow_subset_match

Specifies whether to use subset (token)-based name matching, and if so, what method to use.. Default: “strict”

```
name_match_allow_subset_match
strict| any | none
```

name_match_based_on_nets

Controls whether compare point matching will be based on net names. Default: “true”

```
name_match_based_on_nets true | false
```

name_match_filter_chars

Specifies the characters to be ignored when the name matching filter is used. Default: “`~!@#$%^&*()_-+=|\[]{}'":;?,./”

```
name_match_filter_chars char_list
```

name_match_flattened_hierarchy_separator_style

Specifies the separator used in path names Formality creates when it flattens a design during hierarchical verification. Default: “/”

```
name_match_flattend_hierarchy_separator_style
char
```

name_match_multibit_register_reverse_order

Specifies to reverse the bit order of the bits of a multibit register. Default: “false”

```
name_match_multibit_register_reverse_order
true | false
```

name_match_net

Causes name matching to attempt to match internal nets in the reference to design to internal nets in the implementation design. Default: “false”

```
name_match_pin_net true | false
```

name_match_pin_net

Causes name matching to attempt to match unmatched hierarchical pins in the reference to design to internal nets in the implementation design. Default: “false”

```
name_match_pin_net true | false
```

name_match_use_filter

Specifies whether or not the built-in name matching filter should be used. Default: “true”

```
name_match_use_filter true | false
```

ref

Set to indicate the current reference design. Default: “ ”

```
ref
```

schematic_expand_logic_cone

Specifies whether or not the schematic view of the logic cone displays the internals of techlib cells and DesignWare components. Default: “false”

```
schematic_expand_logic_cone
true | auto | false
```

search_path

Specifies the directories to search for design and library files specified without directory names. Default: “ ”

```
search_path dirlist
```

sh_arch

Set to indicate the current system architecture of the machine you are using.

```
sh_arch
```

sh_continue_on_error

Controls whether processing continues when errors occur. Default: “false”

```
sh_continue_on_error true | false
```

sh_enable_line_editing

Enables the command-line editing capabilities in Formality. Default: “false”

```
sh_enable_line_editing true | false
```

sh_enable_page_mode

Specifies to display long reports one screen at a time. Default: “false”

sh_enable_page_mode true | false

sh_line_editing_mode

Enables vi or emacs editing more in the Formality shell. Default: “emacs”

sh_line_editing_mode vi | emacs

sh_product_version

Specifies the version of the application. Default: “ ”

sh_product_version *string*

sh_source_uses_search_path

Causes the source command to use the search_path variable to search for files. Default: “false”

sh_source_uses_search_path true | false

signature_analysis_allow_net_match

Specifies whether signature analysis uses net-based matching methods. Default: “false”

signature_analysis_allow_net_match
true | false

signature_analysis_allow_subset_match

Specifies whether signature analysis uses subset matching methods. Default: “true”

signature_analysis_allow_subset_match
true | false

signature_analysis_match_blackbox_input

Specifies whether signature analysis attempts to match previously unmatched black box inputs. Default: “true”

signature_analysis_match_blackbox_input
true | false

signature_analysis_match_blackbox_output

Specifies whether signature analysis attempts to match previously unmatched black box outputs. Default: “true”

```
signature_analysis_match_blackbox_output
true | false
```

signature_analysis_match_compare_points

Specifies whether signature analysis attempts to match compare points. Default: “true”

```
signature_analysis_match_compare_points
true | false
```

signature_analysis_match_datapath

Controls whether or not to automatically attempt to match previously unmatched datapath blocks and their pins during signature analysis. Default: “true”

```
signature_analysis_match_datapath
true | false
```

signature_analysis_match_hierarchy

Controls whether or not to automatically attempt to match previously unmatched hierarchical blocks and their pins during signature analysis. Default: “true”

```
signature_analysis_match_hierarchy
true | false
```

signature_analysis_match_net

Causes signature analysis to attempt to match internal nets in the reference design to internal nets in the implementation. Default: “true”

```
signature_analysis_match_net
true | false
```

signature_analysis_match_pin_net

Causes signature analysis to attempt to match unmatched hierarchical pins in the reference design to internal nets in the implementation. Default: “true”

```
signature_analysis_match_pin_net
true | false
```

signature_analysis_match_primary_input

Controls whether or not signature analysis attempts to match previously unmatched primary inputs. Default: “true”

```
signature_analysis_match_primary_input
true | false
```

signature_analysis_match_primary_output

Controls whether or not signature analysis attempts to match previously unmatched primary outputs. Default: “false”

```
signature_analysis_match_primary_output
true | false
```

signature_analysis_matching

Controls whether signature analysis will be used to help match compare points. Default: “true”

```
signature_analysis_matching true | false
```

svf_datapath

Controls whether Formality processes the transformations found in the SVF file. Default: “false”

```
svf_datapath
true | false
```

synopsys_root

Sets the value of the $SYNOPSYS environment variable. Default: “ ”

```
synopsys_root pathname
```

verification_assume_constant_reg_init

Obsolete. Controls how Formality propagates constants through possible constant registers. Default: “true”

```
verification_assume_constant_register_init
true | false
```

verification_assume_reg_init

Controls the assumptions that Formality makes about initial register states. Default: “liberal”

```
verification_assume_register_init
none | conservative | liberal | liberal0 |
liberal1
```

verification_asynch_bypass

Controls the verification of registers having asynchronous controls that operate by creating a combinational “bypass” around the register against registers with asynchronous controls that operate only by setting the register state. Default: “false”

```
verification_asynch_bypass true | false
```

verification_auto_loop_break

Performs simple loop breaking by cutting the nets. Default: “true”

```
verification_auto_loop_break
true | false
```

verification_blackbox_match_mode

Defines how Formality matches comparable black boxes during verification. Default: “any”

```
verification_blackbox_match_mode
any | identity
```

verification_clock_gate_hold_mode

Specifies whether Formality should allow the next state of a flip-flop whose clock is gated to be considered equivalent to that of a flip-flop whose clock is not gated. Default: “none”

verification_clock_gate_hold_mode
none | low | high | any

verification_constant_prop_mode

Specified how Formality propagated constants through flattened levels of design hierarchy during verification. Default: auto

verification_constant_prop_mode
auto | top | target

verification_datapath_effort_level

Defines the effort level Formality applies during datapath block verification. Default: “automatic”

verification_datapath_effort_level
low | medium | high | unlimited | automatic

verification_effort_level

Specifies the effort level used when verifying two designs. Default: “high”

verification_effort_level
low | medium | high

verification_failing_point_limit

Specifies the number of failing compare points identified before Formality halts the verification process. Default: “20”

verification_failing_point_limit *integer*

verification_ignore_unmatched_implementation_blackbox_input

Defines whether Formality allows unmatched implementation blackbox input pins in a succeeding verification. Default: “true”

```
verification_ignore_unmatched_blackbox_implementation true | false
```

verification_incremental_mode

Controls whether the verify command verifies incrementally. Default: “true”

```
verification_incremental_mode
true | false
```

verification_inversion_push

Defines whether Formality attempts to correct failing verifications by checking for cases where data inversion has been moved across register boundaries. Default: “false”

```
verification_inversion_push true | false
```

verification_match_undriven_signals

This variable is obsolete. Use verification_set_undriven_signals instead.

verification_merge_duplicated_registers

Specifies whether Formality should identify and merge duplicated registers. Default: “false”

```
verification_merge_duplicated_registers
true | false
```

verification_parameter_checking

Enables parameter checking for technology library register cells and black box cells. Default: “false”

```
verification_parameter_checking
consistency | false
```

verification_partition_timeout_limit

Causes Formality to terminate processing of any partition after the time limit is reached, saving partial results, if any, but continuing with other partitions. Default: “ ”

```
verification_partition_timeout_limit
hh:mm:ss | ss
```

verification_passing_mode

Sets the verification mode. Default: “consistency”

```
verification_passing_mode
consistency | equality
```

verification_progress_report_interval

Specifies the interval between progress reports during verification, in minutes. Default: “30”

```
verification_progress_report_interval
integer
```

verification_propagate_const_reg_x

Specifies how Formality propagates don't-care states through reconvergent-fanout-dependent constant-disabled registers. Default: “false”

```
verification_propagate_const_reg_x
true | false
```

verification_set_undriven_signals

Specifies how Formality treats undriven nets and pins during verification. Default: “binary:X”

```
verification_set_undriven_signals
X | Z | 0 | 1 | PI | Binary |
Binary:0 | x:0
```

verification_status

Returns the status of the most recent verification, if any. Default: “not run”

```
verification_status
```

verification_super_low_effort_first_pass

Controls whether Formality makes an initial super_low_effort verification attempt on all compare points. Default: “true”

```
verification_super_low_effort_first_pass
true | false
```

verification_timeout_limit

Specifies a wall-clock time limit for how long Formality spends in verification. Default: “none”

```
verification_timeout_limit value
```

verification_use_partial_modeled_cells

Specifies how Formality tests components defined in Synopsys .lib files with "mission mode only" and whether it treats them as black boxes. Default: “false”

```
verification_use_partial_modeled_cells
true | false
```

verification_verify_unread_compare_points

Controls whether or not Formality verifies unread compare points. Default: “false”

```
verification_verify_unread_compare_points
true | false
```

verification_verify_unread_tech_cell_pins

Allows individual unread input pins of tech library cells in the reference design to be matched with the corresponding cells in the implementation design, and treated as primary outputs for verification. Default: “false”

```
verification_verify_unread_tech_cell_pins
true | false
```