Libero SoC Tcl Commands

Reference Guide

NOTE: PDF files are intended to be viewed on the printed page; links and cross-references in this PDF file may point to external files and generate an error when clicked. **View the online help included with software to enable all linked content.**

Microsemi

Table of Contents

Introduction to Tcl Scripting

Tcl, the Tool Command Language, pronounced *tickle*, is an easy-to-learn scripting language that is compatible with Libero SoC and Designer software. You can run scripts from either the Windows or UNIX command line or store and run a series of commands in a *.tcl batch file.

This section provides a quick overview of the main features of Tcl:

For complete information on Tcl scripting, refer to one of the books available on this subject. You can also find information about Tcl at web sites such as http://www.tcl.tk.

- Basic syntax
- Types of Tcl commands
- Variables
- Command substitution
- Quotes and braces
- Lists and arrays
- Control structures
- Handling exceptions
- Print statement and Return values
- Running Tcl scripts from the command line
- Running Tcl scripts from the GUI
- Exporting Tcl scripts
- Extended_run_gui
- Extended_run_shell
- Sample Tcl scripts
- Project Manager Tcl Commands
- Designer Tcl Commands

Tcl Commands and Supported Families

When we specify a family name, we refer to the device family and all its derivatives, unless otherwise specified. See the table below for a list of supported device families and their derivatives.

The table below shows the supported families for each Tcl command.

Note: Entries preceded by a "*" indicate Designer commands that cannot be executed in Libero Tcl scripts.

Note: Entries preceded by a "**" indicate SmartTime commands that cannot be executed in Libero Tcl scripts.

Command	SmartFusion2, IGLOO2, and RTG4	SmartFusion	IGLOO	ProASIC3	Fusion
add_file_to_library	X	X	X	X	X
add_library	X	X	X	X	X
add_modelsim_path	X	X	X	X	X
*add_probe		X	X	X	X
add_profile	X	X	X	X	X

Command	SmartFusion2, IGLOO2, and RTG4	SmartFusion	IGLOO	ProASIC3	Fusion
*all_inputs		X	X	X	X
*all_outputs		X	X	X	X
*all_registers		X	X	X	X
*are_all_source_files_current		X	X	X	X
associate_stimulus	X	X	X	X	X
*backannotate		X	X	X	X
change_link_source	X	X	X	X	X
**check_constraints					
check_hdl	X	X	X	X	X
check_schematic	X	X	X	X	X
*check_timing_constraints		X	X	X	X
*clone_scenario		X	X	X	X
close_design	X	X	X	X	X
close_project	X	X	X	X	X
*compile		X	X	X	X
configure_tool (SmartFusion, IGLOO, ProASIC3 and Fusion)		X	X	X	X
configure_tool (SmartFusion2, IGLOO2, and RTG4)	X				
*create_clock		X	X	X	X
*create_generated_clock		X	X	X	X
create_links	X	X	X	X	X
*create_scenario		X	X	X	X
**create_set					
create_symbol	X	X	X	X	X
defvar_get	X	X	X	X	X
defvar_set	X	X	X	X	X

Command	SmartFusion2, IGLOO2, and RTG4	SmartFusion	IGLOO	ProASIC3	Fusion
delete_files	X	X	X	X	X
*delete_probe		X	X	X	X
*delete_scenario		X	X	X	X
download_core	X	X	X	X	X
edit_profile	X	X	X	X	X
**expand_path					
*export		X	X	X	X
export (Block support)		X	X	X	X
export_as_link	X	X	X	X	X
export_ba_files	X				
export_bitstream_file	X				
export_bsdl_file	X				
export_design_summary	X				
export_firmware	X				
export_fp_pdc	X				
export_io_constraints_from_adb		X	X	X	X
export_io_pdc	X				
export_netlist_file	X				
export_profiles	X	X	X	X	X
export_prog_job	X				
export_script	X	X	X	X	X
export_sdc_file	X				
generate_ba_files		X	X	X	X
generate_hdl_from_schematic	X	X	X	X	X
generate_hdl_netlist		X	X	X	X
*generate_probes		X	X	X	X

Command	SmartFusion2, IGLOO2, and RTG4	SmartFusion	IGLOO	ProASIC3	Fusion
*get_cells		X	X	X	X
*get_clocks		X	X	X	X
*get_current_scenario		X	X	X	X
*get_defvar		X	X	X	X
*get_design_filename		X	X	X	X
*get_design_info		X	X	X	X
*get_nets		X	X	X	X
*get_out_of_date_files		X	X	X	X
*get_pins		X	X	X	X
*get_ports		X	X	X	X
*import_aux		X	X	X	X
import_component_data	X				
import_files	X	X	X	X	X
*import_source		X	X	X	X
*ioadvisor_apply_suggestion		X	X	X	X
*ioadvisor_commit		X	X	X	X
*ioadvisor_restore		X	X	X	X
*ioadvisor_restore_initial_value		X	X	X	X
*ioadvisor_set_outdrive		X	X	X	X
*ioadvisor_set_outputload		X	X	X	X
*ioadvisor_set_slew		X	X	X	X
*is_design_loaded		X	X	X	X
*is_design_modified		X	X	X	X
*is_design_state_complete		X	X	X	X
*is_source_file_current		X	X	X	X
*layout		X	X	X	X

Command	SmartFusion2, IGLOO2, and RTG4	SmartFusion	IGLOO	ProASIC3	Fusion
*list_clock_latencies		X	X	X	X
*list_clock_uncertainties		X	X	X	X
*list_clocks		X	X	X	X
*list_disable_timings		X	X	X	X
*list_false_paths		X	X	X	X
*list_generated_clocks		X	X	X	X
*list_input_delays		X	X	X	X
*list_max_delays		X	X	X	X
*list_min_delays		X	X	X	X
*list_multicycle_paths		X	X	X	X
*list_objects		X	X	X	X
*list_output_delays		X	X	X	X
**list_paths					
*list_scenarios		X	X	X	X
*new_design		X	X	X	X
new_project	X	X	X	X	X
*open_design		X	X	X	X
open_project	X	X	X	X	X
organize_cdbs	X	X	X	X	X
organize_constraints	X	X	X	X	X
organize_sources	X	X	X	X	X
organize_tool_files	X				
*pin_assign		X	X	X	X
*pin_commit		X	X	X	X
*pin_fix		X	X	X	X
*pin_fix_all		X	X	X	X

Command	SmartFusion2, IGLOO2, and RTG4	SmartFusion	IGLOO	ProASIC3	Fusion
*pin_unassign		X	X	X	X
*pin_unassign_all		X	X	X	X
PROGRAMMING_BITSTREAM_SETTINGS (RTG4 only)					
project_settings	X	X	X	X	X
publish_block (SmartFusion2, IGLOO2, and RTG4)	X				
**read_sdc					
refresh	X	X	X	X	X
**remove_all_constraints					
*remove_clock		X	X	X	X
*remove_clock_latency		X	X	X	X
*remove_clock_uncertainty		X	X	X	X
remove_core	X	X	X	X	X
*remove_disable_timing		X	X	X	X
*remove_false_path		X	X	X	X
*remove_generated_clock		X	X	X	X
*remove_input_delay		X	X	X	X
remove_library	X	X	X	X	X
*remove_max_delay		X	X	X	X
*remove_min_delay		X	X	X	X
*remove_multicycle_path		X	X	X	X
*remove_output_delay		X	X	X	X
remove_profile	X	X	X	X	X
**remove_scenario					
**remove_set					
rename_library	X	X	X	X	X

Command	SmartFusion2, IGLOO2, and RTG4	SmartFusion	IGLOO	ProASIC3	Fusion
*rename_scenario		X	X	X	X
*report (Activity and Hazards Power Report)		X	X	X	X
*report (Bottleneck) using SmartTime		X	X	X	X
*report (Cycle Accurate Power Report)		X	X	X	X
*report (Data History)		X	X	X	X
*report (Datasheet) using SmartTime		X	X	X	X
*report (Power Scenario)		X	X	X	X
*report (Power)		X	X	X	X
*report (Timing violations) using SmartTime		X	X	X	X
*report (Timing) using SmartTime	X	X	X	X	X
run_designer		X	X	X	X
run_drc		X	X	X	X
run_simulation	X	X	X	X	X
run_synthesis	X	X	X	X	X
run_tool (SmartFusion, IGLOO, ProASIC3 and Fusion)		X	X	X	X
run_tool (SmartFusion2, IGLOO2, and RTG4)	X				
**save					
*save_design		X	X	X	X
save_log	X	X	X	X	X
save_project	X	X	X	X	X
save_project_as	X	X	X	X	X
select_profile	X	X	X	X	X

Command	SmartFusion2, IGLOO2, and RTG4	SmartFusion	IGLOO	ProASIC3	Fusion
set_actel_lib_options	X	X	X	X	X
*set_clock_latency		X	X	X	X
**set_clock_to_output	X	X	X	X	X
*set_clock_uncertainty		X	X	X	X
*set_current_scenario		X	X	X	X
*set_defvar		X	X	X	X
*set_design		X	X	X	X
*set_device		X	X	X	X
set_device (Project Manager)	X	X	X	X	X
*set_disable_timing		X	X	X	X
**set_external_check	X	X	X	X	X
*set_false_path		X	X	X	X
*set_input_delay		X	X	X	X
*set_max_delay		X	X	X	X
*set_min_delay		X	X	X	X
set_modelsim_options	X	X	X	X	X
*set_multicycle_path		X	X	X	X
set_option		X	X	X	X
**set_options					
*set_output_delay		X	X	X	X
set_root	X	X	X	X	X
set_user_lib_options	X	X	X	X	X
*smartpower_add_new_custom_mode		X	X	X	X
*smartpower_add_new_scenario		X	X	X	X
*smartpower_add_pin_in_domain		X	X	X	X

Command	SmartFusion2, IGLOO2, and RTG4	SmartFusion	IGLOO	ProASIC3	Fusion
*smartpower_battery_settings		X	X	X	X
*smartpower_change_clock_statistics		X	X	X	X
*smartpower_change_setofpin_statistics		X	X	X	X
*smartpower_commit		X	X	X	X
*smartpower_compute_vectorless		X	X	X	X
*smartpower_create_domain		X	X	X	X
*smartpower_edit_custom_mode		X	X	X	X
*smartpower_edit_scenario		X	X	X	X
*smartpower_import_vcd		X	X	X	X
*smartpower_init_do		X	X	X	X
*smartpower_init_set_clocks_options		X	X	X	X
*smartpower_init_set_combinational_options		X	X	X	X
*smartpower_init_set_enables_options		X	X	X	X
*smartpower_init_set_othersets_options		X	X	X	X
*smartpower_init_set_primaryinputs_options		X	X	X	X
*smartpower_init_set_registers_options		X	X	X	X
*smartpower_init_set_set_reset_options		X	X	X	X
*smartpower_init_setofpins_values		X	X	X	X
*smartpower_initialize_clock_with_constraints		X	X	X	X
*smartpower_remove_all_ann		X	X	X	X

Command	SmartFusion2, IGLOO2, and RTG4	SmartFusion	IGLOO	ProASIC3	Fusion
otations					
*smartpower_remove_custom_mode		X	X	X	X
*smartpower_remove_domain		X	X	X	X
*smartpower_remove_file		X	X	X	X
*smartpower_remove_pin_frequency		X	X	X	X
*smartpower_remove_pin_of_domain		X	X	X	X
*smartpower_remove_pin_probability		X	X	X	X
*smartpower_remove_scenario		X	X	X	X
*smartpower_restore		X	X	X	X
*smartpower_set_cooling		X	X	X	X
*smartpower_set_mode_for_analysis		X	X	X	X
*smartpower_set_mode_for_pdpr		X	X	X	X
*smartpower_set_operating_condition		X	X	X	X
*smartpower_set_operating_conditions					
*smartpower_set_pin_frequency		X	X	X	X
*smartpower_set_preference		X	X	X	X
*smartpower_set_process					
*smartpower_set_scenario_for_analysis		X	X	X	X
*smartpower_set_temperature_opcond		X	X	X	X
*smartpower_set_thermalmode		X	X	X	X

Command	SmartFusion2, IGLOO2, and RTG4	SmartFusion	IGLOO	ProASIC3	Fusion
*smartpower_set_voltage_opcond		X	X	X	X
*smartpower_temperature_opcond_set_design_wide		X	X	X	X
*smartpower_temperature_opcond_set_mode_specific		X	X	X	X
*smartpower_voltage_opcond_set_design_wide		X	X	X	X
*smartpower_voltage_opcond_set_mode_specific		X	X	X	X
*st_commit		X	X	X	X
*st_create_set		X	X	X	X
*st_edit_set		X	X	X	X
*st_expand_path		X	X	X	X
*st_list_paths		X	X	X	X
*st_remove_all_constraints		X	X	X	X
*st_remove_set		X	X	X	X
*st_restore		X	X	X	X
*st_set_options		X	X	X	X
*timer_get_clock_actuals		X	X	X	X
*timer_get_clock_constraints		X	X	X	X
*timer_get_maxdelay		X	X	X	X
*timer_get_path		X	X	X	X
*timer_get_path_constraints		X	X	X	X
*timer_remove_all_constraints		X	X	X	X
*timer_remove_stop		X	X	X	X
*timer_restore		X	X	X	X
unlink	X	X	X	X	X
use_file	X	X	X	X	X

Command					
	SmartFusion2, IGLOO2, and RTG4	**SmartFusion**	**IGLOO**	**ProASIC3**	**Fusion**
use_source_file	X	X	X	X	X
**write_sdc					

Tcl Command Documentation Conventions

The following table shows the typographical conventions used for the Tcl command syntax.

Syntax Notation	Description
`command -argument`	Commands and arguments appear in Courier New typeface.
`variable`	*`Variables appear in blue, italic Courier New typeface. You must substitute an appropriate value for the variable.`*
[`-argument`*`value`*] [*`variable`*]+	Optional arguments begin and end with a square bracket with one exception: if the square bracket is followed by a plus sign (+), then users must specify at least one argument. The plus sign (+) indicates that items within the square brackets can be repeated. Do not enter the plus sign character.

Note: All Tcl commands are case sensitive. However, their arguments are not.

Examples

Syntax for the get_defvar command followed by a sample command:

```
get_defvar variable
```

```
get_defvar “DESIGN”
```

Syntax for the backannotate command followed by a sample command:

```
backannotate -name file_name -format format_type -language language -dir directory_name [-netlist] [-pin]
```

```
backannotate -dir \
 {..\design} -name "fanouttest_ba.sdf" -format "SDF" -language "VERILOG" \
-netlist
```

Wildcard Characters

You can use the following wildcard characters in names used in Tcl commands:

Wildcard	What it Does
\	Interprets the next character literally
?	Matches any single character
*	Matches any string

Wildcard	What it Does
[]	Matches any single character among those listed between brackets (that is, [A-Z] matches any single character in the A-to-Z range)

Note: The matching function requires that you add a slash (\) before each slash in the port, instance, or net name when using wildcards in a PDC command and when using wildcards in the Find feature of the MultiView Navigator. For example, if you have an instance named “A/B12” in the netlist, and you enter that name as “A\\/B*” in a PDC command, you will not be able to find it. In this case, you must specify the name as A\\\\/B*.

Special Characters [ ], { }, and \

Sometimes square brackets ([ ]) are part of the command syntax. In these cases, you must either enclose the open and closed square brackets characters with curly brackets ({ }) or precede the open and closed square brackets ([ ]) characters with a backslash (\). If you do not, you will get an error message.

For example:

```
pin_assign -port {LFSR_OUT[0]} -pin 15
or
pin_assign -port LFSR_OUT\[0\] -pin 180
```

Note: Tcl commands are case sensitive. However, their arguments are not.

Entering Arguments on Separate Lines

To enter an argument on a separate line, you must enter a backslash (\) character at the end of the preceding line of the command as shown in the following example:

```
backannotate -dir \
{..\design} -name "fanouttest_ba.sdf" -format "SDF" -language "VERILOG" \
-netlist
```

See Also

Introduction to Tcl scripting

Basic syntax

About Designer Tcl commands

Project Manager Tcl Command Reference

A Tcl (Tool Command Language) file contains scripts for simple or complex tasks. You can run scripts from either the Windows or UNIX command line or store and run a series of Tcl commands in a *.tcl batch file. You can also run scripts from within the GUI in Project Manager.

Note: Tcl commands are case sensitive. However, their arguments are not.

The Libero SoC Project Manager supports the following Tcl scripting commands:

Command	Action
add_file_to_library	Adds a file to a library in your project
add_library	Adds a VHDL library to your project
add_probe	Adds a probe to an internal net in your design, using the original name from the optimized netlist in your design.
add_modelsim_path	Adds a ModelSim simulation library to your

Command	Action
	project
add_profile	Adds a profile; sets the same values as the Add or Edit Profile dialog box
associate_stimulus	Associates a stimulus file in your project
change_link_source	Changes the source of a linked file in your project
check_hdl	Checks the HDL in the specified file
check_schematic	Checks the schematic
close_project	Closes the current project in Libero SoC
configure_tool (SmartFusion2,IGLOO2, and RTG4)	Tcl command to set the parameters for any tool called by Libero for the SmartFusion2, IGLOO2, and RTG4 families.
configure_tool (SmartFusion, IGLOO, ProASIC3, and Fusion)	Tcl command to set the parameters for any tool called by Libero for the SmartFusion, IGLOO, ProASIC3, and Fusion families.
create_links	Creates a link (or links) to a file/files in your project
create_symbol	Creates a symbol in a module
delete_files	Deletes files from your Libero SoC project
download_core	Downloads a core and adds it to your repository
edit_profile	Edits a profile; sets the same values as the Add or Edit Profile dialog box
export_as_link	Exports a file to another directory and links to the file
export_ba_files	Exports the backannotated files.
export_bitstream_file	Configures parameters for your exported bitstream.
export_bsdl_file	Exports the BSDL to a specified file.
export_design_summary	Exports an HTML file containing information about your root SmartDesign in your project.
export_firmware	Exports design firmware configuration data.
export fp_pdc	Exports the Floorplanning Physical Design Constraint (*.pdc) File.

Command	Action
export_io_constraints_from_adb	Exports the I/O constraints from your project ADB file to an output file.
export_io_pdc	Exports the I/O constraints Physical Design Constraint (*.pdc) File.
export_netlist_file	Exports the netlist after the compile state has completed.
export_profiles	Exports your tool profiles; performs the same action as the Export Profiles dialog box
export_prog_job	Exports your programming job.
export_script	Explicitly exports the Tcl command equivalents of the current Libero session.
export_sdc_file	Exports the SDC (Synopsys Design Constraint) file for timing constraints.
generate_ba_files	Generates the back-annotate files for your desig.n
generate_hdl_from_schematic	Generates an HDL file from your schematic.
generate_hdl_netlist	Generates the HDL netlist for your design and runs the design rule check.
import_component_data	Imports component data into an existing Libero project.
import_files (Libero SoC)	Imports files into your Libero SoC project
import_vcd	Imports a VCD file into your project
new_project	Creates a new project in the Libero SoC
open_project	Opens an existing Libero SoC project
organize_cdbs	Organizes the CDB files in your project
organize_constraints	Organizes the constraint files in your project
organize_sources	Organizes the source files in your project
organize_tool_files	Specifies specific constraint files to be passed to and used by a Libero tool.
project_settings	Modifies project flow settings for your Libero SoC project
refresh	Refreshes your project, updates the view and checks for updated links and files.
remove_core	Removes a core from your project

Command	Action
remove_library	Removes a VHDL library from your project
remove_pin_enable_rate	Remove a pin enable rate
remove_profile	Deletes a tool profile
rename_library	Renames a VHDL library in your project
rollback_constraints_from_adb	Opens the ADB file, exports the PDC file, and then replaces it with the specified PDC file
run_designer	Runs Designer with compile and layout options (if selected)
run_drc	Runs the design rule check on your netlist and generates an HDL file
run_simulation	Runs simulation on your project with your default simulation tool and creates a logfile
run_tool (SmartFusion2, IGLOO2, and RTG4)	Starts the specified tool.
run_tool (SmartFusion, IGLOO, ProASIC3, and Fusion)	Starts the specified tool.
run_synthesis	Runs synthesis on your project and creates a logfile
save_log	Saves your Libero SoC log file
save_project	Saves your project
save_project_as	Saves your project with a different name
select_profile	Selects a profile to use in your project
set_actel_lib_options	Sets your simulation library to default, or to another library
set_device (Project Manager)	Sets your device family, die, and package in the Project Manager
set_live_probe	Channels A and/or B to the specified probe point(s).
set_modelsim_options	Sets your ModelSim simulation options
set_option	Sets your synthesis options on a module
set_userlib_options	Sets your user library options during simulation
set_root	Sets the module you specify as the root

Command	Action
synplify	Runs Synplify in batch mode and executes a Tcl script.
synplify_pro	Runs Synplify Pro in batch mode and executes a Tcl script.
unlink	Removes a link to a file in your project
use_file	Specifies which file in your project to use
use_source_file	Defines a module for your project

Designer Tcl Command Reference

A Tcl (Tool Command Language) file contains scripts for simple or complex tasks. You can run scripts from either the Windows or UNIX command line or store and run a series of Tcl commands in a ".tcl" batch file. You can also run scripts from within Designer.

Designer supports the following Tcl scripting commands:

Command	Action
add_probe	Adds a probe to an internal net in your design, using the original name from the optimized netlist in your design. Also, this command must be used in conjunction with the generate_probes command to generate a probed ADB file (see example below).
all_inputs	Returns an object representing all input and inout pins in the current design
all_outputs	Returns an object representing all output and inout pins in the current design
all_registers	Returns an object representing register pins or cells in the current scenario based on the given parameters
are_all_source_files_current	Audits all source files and determines whether or not they are out of date / imported into the workspace
backannotate	Extracts timing delays from your post layout data
check_constraints	Checks all timing constraints in the current scenario for validity

Command	Action
check_timing_constraints	Checks all timing constraints in the current timing scenario for validity
clone_scenario	Creates a new timing scenario by duplicating an existing one
close_design	Closes the current design
compile	Performs design rule check and optimizes the input netlist before translating the source code into machine code
create_clock	Creates a clock constraint on the specified ports/pins, or a virtual clock if no source is specified
create_generated_clock	Creates an internally generated clock constraint on the ports/pins and defines its characteristics
create_scenario	Creates a new timing scenario with the specified name
create_set	Creates a set of timing paths
delete_probe	Deletes a probe on nets in a probed ADB file
delete_scenario	Deletes the specified timing scenario
expand_path	Returns the details of a timing path
export	Converts a file from its current format into the specified file format, usually for use in another program
extended_run_shell	Runs multiple iterations of layout through Designer
generate_probes	Executes the probing and creates a new ADB file. This command is used in conjunction with the add_probe Tcl command (see example below).
get_cells	Returns an object representing the cells (instances) that match those specified in the pattern argument
get_clocks	Returns an object representing the clock(s) that match those specified in the pattern argument in the current timing scenario

Command	Action
get_current_scenario	Returns the name of the current timing scenario
get_defvar	Returns the value of the Designer internal variable you specify
get_design_filename	Returns the fully qualified path of the specified design file
get_design_info	Returns detailed information about your design, depending on which arguments you specify
get_nets	Returns an object representing the nets that match those specified in the pattern argument
get_out_of_date_files	Audits all files returns a list of filenames that are out of date
get_pins	Returns an object representing the pin(s) that match those specified in the pattern argument
get_ports	Returns an object representing the port(s) that match those specified in the pattern argument
import_aux	Imports the specified file as an auxiliary file, which are not audited and do not require you to re-compile the design
import_source	Imports the specified file as a source file, which include your netlist and design constraints
ioadvisor_apply_suggestion	Applies the suggestions for the selected attribute to the selected I/O(s)
ioadvisor_commit	Saves all changes in the I/O Advisor
ioadvisor_restore	Restores the I/O Advisor to the initial state
ioadvisor_restore_initial_value	Sets the current value for the selected attribute and I/Os to the initial value
ioadvisor_set_outdrive	Sets the outdrive for the selected I/Os
ioadvisor_set_outputload	Sets the output load for the selected I/Os

Command	Action
ioadvisor_set_slew	Sets the slew for the selected I/Os
is_design_loaded	Returns True if the design is loaded into Designer; otherwise, returns False
is_design_modified	Returns True if the design has been modified since it was last compiled; otherwise, returns False
is_design_state_complete	Returns True if the specified design state is complete (for example, you can inquire as to whether a die and package has been selected for the design); otherwise, returns False
is_source_file_current	Audits the source file and determines whether or not the file is out of date / imported into the workspace
layout	Place-and-route your design
list_clocks	Returns details about all of the clock constraints in the current timing constraint scenario
list_clock_latencies	Returns details about all of the clock latencies in the current timing constraint scenario
list_clock_uncertainties	Returns the list of clock-to-clock uncertainty constraints for the current scenario.
list_disable_timings	Returns the list of disable timing constraints for the current scenario
list_false_paths	Returns details about all of the false paths in the current timing constraint scenario
list_generated_clocks	Returns details about all of the generated clock constraints in the current timing constraint scenario
list_input_delays	Returns details about all of the input delay constraints in the current timing constraint scenario
list_max_delays	Returns details about all of the maximum delay constraints in the current timing constraint scenario
list_min_delays	Returns details about all of the

Command	Action
	minimum delay constraints in the current timing constraint scenario
list_multicycle_paths	Returns details about all of the multicycle paths in the current timing constraint scenario
list_objects	Returns a list of names of the objects in the specified list
list_output_delays	Returns details about all of the output delay constraints in the current timing constraint scenario
list_paths	Returns a list of the n worst paths matching the arguments
list_scenarios	Returns a list of names of all of the available timing scenarios
new_design	Creates a new design (.adb) file in a specific location for a particular design family such ProASIC3
open_design	Opens an existing design in the Designer software
pin_assign	Assigns the named pin to the specified port but does not lock its assignment.
pin_commit	Saves the pin assignments to the design (*.adb) file.
pin_fix	Locks the pin assignment for the specified port, so the pin cannot be moved during place-and-route.
pin_fix_all	Locks all the assigned pins on the device so they cannot be moved during place-and-route.
pin_unassign	Unassigns a specific pin from a specific port. The unassigned pin location is then available for other ports.
pin_unassign_all	Unassigns all pins from a specific port.
pin_unfix	Unlocks the specified pin from its port.
read_sdc	Evaluates an SDC file

Command	Action
remove_all_constraints	Removes all timing constraints
remove_clock	Removes the specified clock constraint from the current timing scenario
remove_clock_latency	Removes a clock source latency from the specified clock and from all edges of the clock
remove_clock_uncertainty	Removes a clock-to-clock uncertainty from the current timing scenario by specifying either its exact arguments or its ID
remove_disable_timing	Removes a disable timing constraint by specifying its arguments, or its ID
remove_false_path	Removes a false path from the current timing scenario by specifying either its exact arguments or its ID
remove_generated_clock	Removes the specified generated clock constraint from the current scenario
remove_input_delay	Removes an input delay a clock on a port by specifying both the clocks and port names or the ID of the input_delay constraint to remove
remove_max_delay	Removes a maximum delay constraint in the current timing scenario by specifying either its exact arguments or its ID.
remove_min_delay	Removes a minimum delay constraint in the current timing scenario by specifying either its exact arguments or its ID
remove_multicycle_path	Removes a multicycle path constraint in the current timing scenario by specifying either its exact arguments or its ID
remove_output_delay	Removes an ouput delay by specifying both the clocks and port names or the ID of the output_delay constraint to remove
remove_scenario	Removes a scenario from the constraint database

Command	Action
rename_scenario	Renames the specified timing scenario with the new name provided
remove_set	Removes a set of user-created timing paths
report	Generates the type of report you specify: Status, Timing, Timer Violations, Flip-flop, Power, Pin, or I/O Bank
report (Activity and Hazards Power Report)	Reads a VCD file and reports transitions and hazards for each clock cycle of the VCD file.
report (Bottleneck) using SmartTime	Creates a bottleneck report
report (Cycle Accurate Power Report)	Reports a power waveform with one power value per clock period or half-period instead of an average power for the whole simulation
Report (Data History)	Reports new features and enhancements, bug fixes and known issues for the current release that may impact the power consumption of the design
report (Datasheet) using SmartTime	Creates a datasheet report
Report (Power)	Creates a Power report, which enables you to determine if you have any power consumption problems in your design
Report (Power Scenario)	Creates a scenario power report, which enables you to enter a duration for a sequence of previously defined power modes and calculate the average power consumption and the excepted battery life for this sequence.
report (Timing) using SmartTime	Creates a timing report
report (Timing violations) using SmartTime	Creates a timing violations report
set_clock_latency	Defines the delay between an external clock source and the definition pin of a clock within SmartTime
set_clock_uncertainty	Specifies a clock-to-clock uncertainty and returns the ID of the

Command	Action
	created constraint if the command succeeded
set_current_scenario	Specifies the timing scenario for the Timing Analyzer to use
save_design	Writes the design to the specified filename
set_defvar	Sets the value of the Designer internal variable you specify >
set_design	Specifies the design name, family and path in which Designer will process the design
set_device	Specifies the type of device and its parameters
set_disable_timing	Disables timing arcs within a cell and returns the ID of the created constraint
set_false_path	Identifies paths that are considered false and excluded from the timing analysis in the current timing scenario
set_input_delay	Creates an input delay on a port list by defining the arrival time of an input relative to a clock in the current scenario
set_max_delay	Specifies the maximum delay for the timing paths in the current scenario
set_min_delay	Specifies the minimum delay for the timing paths in the current scenario
set_multicycle_path	Defines a path that takes multiple clock cycles in the current scenario
set_output_delay	Defines the output delay of an output relative to a clock in the current scenario
smartpower_add_new_custom_mode	Creates a new custom mode
smartpower_add_new_scenario	Creates a new scenario
smartpower_add_pin_in_domain	Adds a pin to either a Clock or Set domain
smartpower_battery_settings	Sets the battery capacity in SmartPower

Command	Action
smartpower_change_clock_statistics	Changes the default frequencies and probabilities for a specific domain
smartpower_change_setofpin_statistics	Changes the default frequencies and probabilities for a specific set
smartpower_commit	Saves the changes made in SmartPower to the design file (.adb) in Designer
smartpower_compute_vectorless	Executes a vectorless analysis of the current operating mode
smartpower_create_domain	Creates a new clock or set domain
smartpower_edit_custom_mode	Edits a custom mode
smartpower_edit_scenario	Edits a scenario
smartpower_import_vcd	Imports into SmartPower a VCD file generated by a simulation tool
smartpower_initialize_clock_with_constraints	Initializes the clock frequency and the data frequency of a single clock domain with a specified clock name and the initialization options
smartpower_init_do	Initializes the frequencies and probabilities for clocks, registers, set/reset nets, primary inputs, combinational outputs, enables and other sets of pins, and selects a mode for initialization
smartpower_init_set_clocks_options	Initializes the clock frequency of all clock domains
smartpower_init_set_combinational_options	Initializes the frequency and probability of all combinational outputs
smartpower_init_set_enables_options	Initializes the clock frequency of all enable clocks with the initialization options
smartpower_init_set_othersets_options	Initializes the frequency and probability of all other sets
smartpower_init_set_primaryinputs_options	Initializes the frequency and probability of all primary inputs
smartpower_init_set_registers_options	Initializes the frequency and probability of all register outputs

Command	Action
smartpower_init_set_set_reset_options	Initializes the frequency and probability of all set/reset nets
smartpower_init_setofpins_values	Initializes the frequency and probability of all sets of pins
smartpower_remove_all_annotations	Removes all initialization annotations for the specified mode
smartpower_remove_custom_mode	Removes a custom mode
smartpower_remove_domain	Removes an existing domain
smartpower_remove_file	Removes a VCD file from the specified mode
smartpower_remove_pin_enable_rate	This command is obsolete and it is replaced by smartpower_remove_pin_probability
smartpower_remove_pin_frequency	Removes the frequency of an existing pin
smartpower_remove_pin_of_domain	Removes a clock pin or a data pin from a Clock or Set domain, respectively.
smartpower_remove_pin_probability	Enables you to annotate the probability of a pin driving an enable pin
smartpower_remove_scenario	Removes a scenario from the current design
smartpower_remove_vcd	Removes an existing VCD file from a mode or entire design
smartpower_restore	Restores previously committed constraints
smartpower_set_battery_capacity	Sets the battery capacity
smartpower_set_cooling	Sets the cooling style to one of the predefined types, or a custom value
smartpower_set_mode_for_analysis	Sets the mode for cycle-accurate power analysis
smartpower_set_mode_for_pdpr	Sets the operating mode used by the Power Driven Place and Route (PDPR) tool during power optimization
smartpower_set_operating_condition	Sets the operating conditions used in SmartPower to best, typical, or

Command	Action
	worst case
smartpower_set_pin_enable_rate	This command is obsolete and it is now replaced by smartpower_set_pin_probability
smartpower_set_pin_frequency	Sets the frequency of an existing pin
smartpower_set_pin_probability	Enables you to annotate the probability of a pin driving an enable pin
smartpower_set_preferences	Sets SmartPower preferences such as power unit, frequency unit, operating mode, operating conditions, and toggle
smartpower_set_scenario_for_analysis	Sets the scenario for cycle-accurate power analysis
smartpower_set_temperature_opcond	Sets the temperature in the operating conditions used in SmartPower
smartpower_set_thermalmode	Sets the mode of computing junction temperature
smartpower_set_voltage_opcond	Sets the voltage in the operating conditions used in SmartPower
smartpower_temperature_opcond_set_design_wide	Sets the temperature for SmartPower design-wide operating conditions
smartpower_temperature_opcond_set_mode_specific	Sets the temperature for SmartPower mode-specific operating conditions
smartpower_voltage_opcond_set_design_wide	Sets the voltage settings for SmartPower design-wide operating conditions
smartpower_voltage_opcond_set_mode_specific	Sets the voltage settings for SmartPower mode-specific use operating conditions
st_create_set	Creates a set of paths to be analyzed
st_commit	Saves the changes made in SmartTime to the design (.adb) file
st_edit_set	Modifies the paths in a user set
st_expand_path	Displays expanded path information

Command	Action
	(path details) for paths
st_list_paths	Displays the list of paths in the same tabular format shown in SmartTime
st_remove_all_constraints	Removes all timing constraints
st_remove_set	Deletes a user set from the design
st_restore	Restores constraints previously committed in SmartTime
st_set_options	Sets options for timing analysis
timer_get_path	Displays the Timer path information in the Log window
timer_get_clock_actuals	Displays the actual clock frequency in the Log window
timer_get_clock_constraints	Displays the clock constraints (period/frequency and dutycycle) in the Log window
timer_get_maxdelay	Displays the maximum delay constraint between two pins of a path in the Log window
timer_get_path_constraints	Displays the path constraints set for maxdelay in the Timer in the Log window
timer_remove_stop	Removes the path stop constraint on the specified pin
timer_restore	Restores previously committed constraints
timer_remove_all_constraints	Removes all the timing constraints previously entered in the Designer system
write_sdc	Writes timing constraints into an SDC file

Note: Tcl commands are case sensitive. However, their arguments are not.

See Also

Introduction to Tcl scripting

Basic syntax

Basic Syntax

Tcl scripts contain one or more commands separated by either new lines or semicolons. A Tcl command consists of the name of the command followed by one or more arguments. The format of a Tcl command is:

```
command arg1 ... argN
```

The command in the following example computes the sum of 2 plus 2 and returns the result, 4.

```
expr 2 + 2
```

The **expr** command handles its arguments as an arithmetic expression, computing and returning the result as a string. All Tcl commands return results. If a command has no result to return, it returns an empty string.

To continue a command on another line, enter a backslash (\) character at the end of the line. For example, the following Tcl command appears on two lines:

```
import -format "edif" -netlist_naming "Generic" -edif_flavor "GENERIC" {prepi.edn}
```

Comments must be preceded by a hash character (#). The comment delimiter (#) must be the first character on a line or the first character following a semicolon, which also indicates the start of a new line. To create a multi-line comment, you must put a hash character (#) at the beginning of each line.

Note: Be sure that the previous line does not end with a continuation character (\). Otherwise, the comment line following it will be ignored.

Special Characters

Square brackets ([ ]) are special characters in Tcl. To use square brackets in names such as port names, you must either enclose the entire port name in curly braces, for example, pin_assign -port {LFSR_OUT[15]} -iostd lvttl -slew High, or lead the square brackets with a slash (\) character as shown in the following example:
pin_assign -port LFSR_OUT\[15\] -iostd lvttl -slew High

Sample Tcl Script

```
#Set up a new design
new_design -name "multiclk" -family "Axcelerator" -path {.}

# Set device, package, speed grade, default I/O standard and
# operating conditions
set_device -die "AX1000" -package "BG729" -speed "-3" \
-voltage "1.5" -iostd "LVTTL" -temprange "COM" -voltrange "COM"

# Import the netlist
import -format "verilog" {multiclk.v}

# Compile the netlist
compile

# Import a PDC file
import_aux -format "pdc" {multiclk.pdc}

# Run standard layout
layout -incremental "OFF"

# Generate backannotated sdf and netlist file
backannotate -name {multiclk_ba} -format "sdf" -language "Verilog"

# Generate timing report
report -type "timing" -sortby "actual" -maxpaths "100" {report_timing.txt}
```

```
# Generate programming file
export -format "AFM" -signature "ffff" {multiclk.afm}
```

Types of Tcl commands

There are three types of Tcl commands:

- Built-in commands
- Procedures created with the proc command
- Commands built into the Designer software

Built-in commands

Built-in commands are provided by the Tcl interpreter. They are available in all Tcl applications. Here are some examples of built-in Tcl commands:

- Tcl provides several commands for manipulating file names, reading and writing file attributes, copying files, deleting files, creating directories, and so on.
- exec - run an external program. Its return value is the output (on stdout) from the program, for example:

```
set tmp [ exec myprog ]
puts stdout $tmp
```

- You can easily create collections of values (lists) and manipulate them in a variety of ways.
- You can create arrays - structured values consisting of name-value pairs with arbitrary string values for the names and values.
- You can manipulate the time and date variables.
- You can write scripts that can wait for certain events to occur, such as an elapsed time or the availability of input data on a network socket.

Procedures created with the proc command

You use the proc command to declare a procedure. You can then use the name of the procedure as a Tcl command.

The following sample script consists of a single command named **proc**. The proc command takes three arguments:

- The name of a procedure (myproc)
- A list of argument names (arg1 arg2)
- The body of the procedure, which is a Tcl script

```
proc myproc { arg1 arg2 } {
# procedure body
}
myproc a b
```

Commands built into the software

Many functions that you can perform through the software's GUI interface, you can also perform using an equivalent Tcl command. For example, the `backannotate` command is equivalent to executing the Back-Annotate command from Designer's Tools menu. For a list of Tcl commands supported in the Designer software, see "Tcl Commands."

Variables

With Tcl scripting, you can store a value in a variable for later use. You use the set command to assign variables. For example, the following set command creates a variable named x and sets its initial value to 10.

```
set x 10
```

A variable can be a letter, a digit, an underscore, or any combination of letters, digits, and underscore characters. All variable values are stored as strings.

In the Tcl language, you do not declare variables or their types. Any variable can hold any value. Use the dollar sign ($) to obtain the value of a variable, for example:

```
set a 1
set b $a
set cmd expr
set x 11
$cmd $x*$x
```

The dollar sign $ tells Tcl to handle the letters and digits following it as a variable name and to substitute the variable name with its value.

Global Variables

Variables can be declared global in scope using the Tcl global command. All procedures, including the declaration can access and modify global variables, for example:

```
global myvar
```

Command substitution

By using square brackets ([]), you can substitute the result of one command as an argument to a subsequent command, as shown in the following example:

```
set a 12
set b [expr $a*4]
```

Tcl handles everything between square brackets as a nested Tcl command. Tcl evaluates the nested command and substitutes its result in place of the bracketed text. In the example above, the argument that appears in square brackets in the second set command is equal to 48 (that is, 12* 4 = 48).

Conceptually,

```
set b [expr $a * 4]
```

expands to

```
set b [expr 12 * 4 ]
```

and then to

```
set b 48
```

Quotes and braces

The distinction between braces ({ }) and quotes (" ") is significant when the list contains references to variables. When references are enclosed in quotes, they are substituted with values. However, when references are enclosed in braces, they are not substituted with values.

Example

With Braces	With Double Quotes
set b 2	set b 2
set t { 1 $b 3 }	set t " 1 $b 3 "
set s { [expr $b + $b] }	set s " [expr $b + $b] "
puts stdout $t	puts stdout $t
puts stdout $s	puts stdout $s

will output

```
1 $b 3                          vs.          1 2 3
[ expr $b + $b ]                             4
```

Filenames

In Tcl syntax, filenames should be enclosed in braces { } to avoid backslash substitution and white space separation. Backslashes are used to separate folder names in Windows-based filenames. The problem is that sequences of “\n” or “\t” are interpreted specially. Using the braces disables this special interpretation and specifies that the Tcl interpreter handle the enclosed string literally. Alternatively, double-backslash “\\n” and “\\t” would work as well as forward slash directory separators “/n” and “/t”.For example, to specify a file on your Windows PC at c:\newfiles\thisfile.adb, use one of the following:

```
{C:\newfiles\thisfile.adb}
C:\\newfiles\\thisfile.adb
"C:\\newfiles\\thisfile.adb"
C:/newfiles/thisfile.adb
"C:/newfiles/thisfile.adb"
```

If there is white space in the filename path, you must use either the braces or double-quotes. For example:

```
C:\program data\thisfile.adb
```

should be referenced in Tcl script as

```
{C:\program data\thisfile.adb} or "C:\\program data\\thisfile.adb"
```

If you are using variables, you cannot use braces { } because, by default, the braces turn off all special interpretation, including the dollar sign character. Instead, use either double-backslashes or forward slashes with double quotes. For example:

```
"$design_name.adb"
```

Note: To use a name with special characters such as square brackets [ ], you must put the entire name between curly braces { } or put a slash character \ immediately before each square bracket.

The following example shows a port name enclosed with curly braces:

```
pin_assign -port {LFSR_OUT[15]} -iostd lvttl -slew High
```

The next example shows each square bracket preceded by a slash:

```
pin_assign -port LFSR_OUT\[15\] -iostd lvttl -slew High
```

Lists and arrays

A list is a way to group data and handle the group as a single entity. To define a list, use curly braces { } and double quotes “ “. For example, the following set command {1 2 3 }, when followed by the list command, creates a list stored in the variable "a." This list will contain the items "1," "2," and "3."

```
set a { 1 2 3 }
```

Here's another example:

```
set e 2
set f 3
set a [ list b c d [ expr $e + $f ] ]
puts $a
```

displays (or outputs):

```
b c d 5
```

Tcl supports many other list-related commands such as lindex, linsert, llength, lrange, and lappend. For more information, refer to one of the books or web sites available on this subject.

Arrays

An array is another way to group data. Arrays are collections of items stored in variables. Each item has a unique address that you use to access it. You do not need to declare them nor specify their size.

Array elements are handled in the same way as other Tcl variables. You create them with the set command, and you can use the dollar sign ($) for their values.

```
set myarray(0) "Zero"
set myarray(1) "One"
set myarray(2) "Two"
for {set i 0} {$i < 3} {incr i 1} {
```

Output:

```
Zero
One
Two
```

In the example above, an array called "myarray" is created by the set statement that assigns a value to its first element. The for-loop statement prints out the value stored in each element of the array.

Special arguments (command-line parameters)

You can determine the name of the Tcl script file while executing the Tcl script by referring to the $argv0 variable.

```
puts “Executing file $argv0”
```

To access other arguments from the command line, you can use the `lindex` command and the *`argv`* variable:

To read the the Tcl file name:

```
lindex $argv 0
```

To read the first passed argument:

```
lindex $argv 1
```

Example

```
puts "Script name is $argv0" ; # accessing the scriptname
puts "first argument is [lindex $argv 0]"
puts "second argument is [lindex $argv 1]"
puts "third argument is [lindex $argv 2]"
puts "number of argument is [llength $argv]"
set des_name [lindex $argv 0]
puts "Design name is $des_name"
```

Control structures

Tcl control structures are commands that change the flow of execution through a script. These control structures include commands for conditional execution (if-then-elseif-else) and looping (while, for, catch).

An "if" statement only executes the body of the statement (enclosed between curly braces) if the Boolean condition is found to be true.

if/else statements

```
if { “$name” == “paul” } then {
…
# body if name is paul
} elseif { $code == 0 } then {
…
# body if name is not paul and if value of variable code is zero
} else {
…
# body if above conditions is not true
}
```

for loop statement

A "for" statement will repeatedly execute the body of the code as long as the index is within a specified limit.

```
for { set i 0 } { $i < 5 } { incr i } {
…
# body here
}
```

while loop statement

A "while" statement will repeatedly execute the body of the code (enclosed between the curly braces) as long as the Boolean condition is found to be true.

```
while { $p > 0 } {
…
}
```

catch statement

A "catch" statement suspends normal error handling on the enclosed Tcl command. If a variable name is also used, then the return value of the enclosed Tcl command is stored in the variable.

```
catch { open “$inputFile” r } myresult
```

Handling Exceptions (Tcl Scripting)

To control the flow of the Designer software based on certain conditions (for example, success or failure of certain commands), you can use the Tcl built-in catch command as follows:

```
if { [ catch {open_design $des_name.adb} ] } {
    puts "Cannot open $des_name.adb"
    export -format "log" -diagnostic $des_name.log"
   return 1
    } else {
   puts "Design $des_name.adb Successfully Opened"
}
## set layout mode to standard
layout -incremental "OFF"
if { [ catch {layout} ] } {
    puts "Layout Failed"
    export -format "log" -diagnostic $des_name.log"
    return 1
} else {
   puts "layout successful"
  export -format log "$des_name.log"
  save_design "$des_name.adb";
  close_design
}
```

Print statement and Return values

Print Statement

Use the puts command to write a string to an output channel. Predefined output channels are “stdout” and “stderr.” If you do not specify a channel, then puts display text to the stdout channel.

Note: The STDIN Tcl command is not supported by Microsemi SoC tools.

Example:

```
set a [ myprog arg1 arg2 ]
puts "the answer from myprog was $a (this text is on stdout)"
puts stdout “this text also is on stdout”
```

Return Values

The return code of a Tcl command is a string. You can use a return value as an argument to another function by enclosing the command with square brackets [ ].

Example:

```
set a [ prog arg1 arg2 ]
exec $a
```

The Tcl command “exec” will run an external program. The return value of “exec” is the output (on stdout) from the program.

```
Example:
set tmp [ exec myprog ]
puts stdout $tmp
```

Running Tcl Scripts from the GUI

Instead of running scripts from the command line, you can use Execute Script dialog box to run a script in the software.

To run a Tcl script from the GUI:

1. In Libero SoC, from the **File** menu choose **Execute Script**.

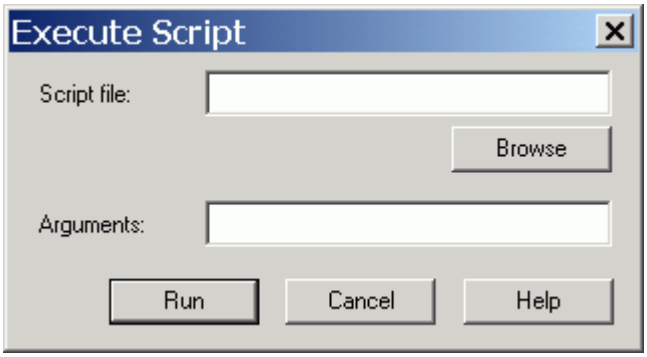

Figure 1 · Execute Script Dialog Box

2. Click **Browse** to display the **Open** dialog box, in which you can navigate to the folder containing the script file to open. When you click **Open**, the software enters the full path and script filename into the Execute Script dialog box for you.
3. In the Arguments edit box, enter the arguments to pass to your Tcl script as shown in the following sample Execute Script dialog box. Separate each argument by a space character. For information about accessing arguments passed to a Tcl script, see "Running Scripts from the command line."

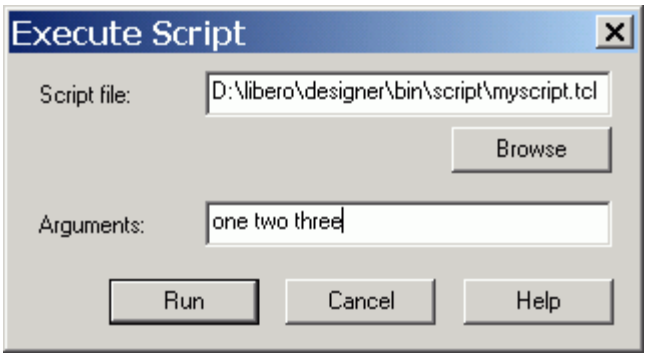

Figure 3 · Execute Script Dialog Box Example

4. Click **Run**.

Specify your arguments in the Execute Script dialog box. To get those argument values from your Tcl script, use the following:

```
puts "Script name: $argv0"
puts "Number of arguments: $argc"
set i 0
foreach arg $argv {
  puts "Arg $i : $arg"
  incr i
}
```

Running Tcl scripts from the Command Line

You can run Tcl scripts from your Windows or Unix command line as well as pass arguments to scripts from the command line.

To execute a Tcl script file in the Libero SoC Project Manager software from a shell command line:

At the prompt, type the path to the Microsemi SoC software followed by the word "SCRIPT" and a colon, and then the name of the script file as follows:

```
<location of Microsemi SoC software>\bin\libero SCRIPT:<filename>
```

where <location of Microsemi SoC software> is the root directory in which you installed the Microsemi SoC software, and <filename> is the name, including a relative or full path, of the Tcl script file to execute. For example, to run the Tcl script file "myscript.tcl", type:

```
C:\libero\designer\bin\libero SCRIPT:myscript.tcl
```

If `myscript.tcl` is in a particular folder named "mydesign", you can use SCRIPT_DIR to change the current working directory before calling the script, as in the following example:

```
C:\libero\designer\bin\libero SCRIPT:myscript.tcl "SCRIPT_DIR:C:\actelprj\mydesign"
```

To execute a Tcl script file in the Designer software from a shell command line:

At the prompt, type the path to the Microsemi SoC software followed by the word "SCRIPT" and a colon, and then the name of the script file as follows:

```
<location of Microsemi SoC software>\bin\designer SCRIPT:<filename>
```

where `<location of Microsemi SoC software>` is the root directory in which you installed the Microsemi SoC software, and`<filename>`is the name, including a relative or full path, of the Tcl script file to execute.

```
For example, to run the Tcl script file named "myscript.tcl" from the command line, you
can type:
C:\libero\designer\bin\designer SCRIPT:myscript.tcl
```

If `myscript.tcl` is in a particular folder named "mydesign", you can use SCRIPT_DIR to change the current working directory before calling the script, as in the following example:

```
C:\libero\designer\bin\designer SCRIPT:myscript.tcl "SCRIPT_DIR:C:\actelprj\mydesign"
```

To pass arguments from the command line to your Tcl script file:

At the prompt, type the path to the Microsemi SoC software followed by the SCRIPT argument. Enclose the entire argument expression in double quotes:

```
<location of Microsemi SoC software>\bin\designer "SCRIPT:<filename arg1 arg2 ...>"
```

where `<location of Microsemi SoC software>` is the root directory in which you installed the Microsemi SoC software, and `<filename arg1 arg2 ...>`is the name, including a relative or full path, of the Tcl script file and arguments you are passing to the script file.

For example,

```
C:\libero\designer\bin\designer "SCRIPT:myscript.tcl  one two three"
```

To obtain the output from the log file:

At the prompt, type the path to the Microsemi SoC software followed by the SCRIPT and LOGFILE arguments.

```
<location of Microsemi SoC software> SCRIPT:<filename> SCRIPT_ARGS:"a b c"
LOGFILE:<output.log>
```

where

- `location of Microsemi SoC software` is the root directory in which you installed the Microsemi SoC software
- `filename` is the name, including a relative or full path, of the Tcl script file
- `SCRIPT_ARGS` are the arguments you are passing to the script file
- `output.log` is the name of the log file

For example,

```
C:\libero\designer\bin\designer SCRIPT:testTCLparam.tcl SCRIPT_ARGS:"a b c"
LOGFILE:testTCLparam.log
```

Exporting Tcl Scripts

You can write out a Tcl script file that contains the commands executed in the current session. You can then use this exported Tcl script to re-execute the same commands interactively or in batch. You can also use this exported script to become more familiar with Tcl syntax.

You can export Tcl scripts from the Project Manager or Designer; the actions are the same.

To export a Tcl session script from the Project Manager or Designer:

1. From the **File** menu, choose **Export Script File**. The **Export Script** dialog box appears.
2. Click **OK**. The **Script Export Options** dialog box appears

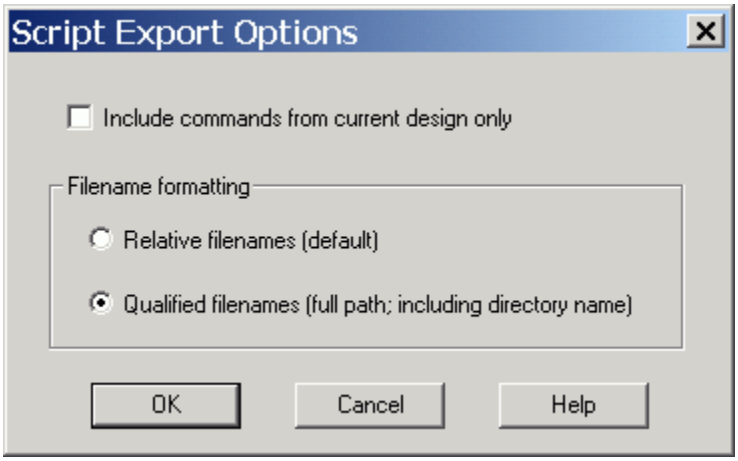

Figure 4 · Script Export Options

5. Check the **Include Commands from Current Design [Project] Only** checkbox. This option applies only if you opened more than one design or project in your current session. If so, and you do not check this box, Project Manager / Designer exports all commands from your current session.

6. Select the radio button for the appropriate filename formatting. To export filenames relative to the current working directory, select **Relative filenames (default)** formatting. To export filenames that include a fully specified path, select **Qualified filenames (full path; including directory name)** formatting.

   Choose **Relative filenames** if you do not intend to move the Tcl script from the saved location, or **Qualified filenames** if you plan to move the Tcl script to another directory or machine.
7. Click **OK**.

Project Manager / Designer saves the Tcl script with the specified filename.

Note:

- When exporting Tcl scripts, Project Manager and Designer always encloses filenames in curly braces to ensure portability.
- Libero SoC software does not write out any Tcl variables or flow-control statements to the exported Tcl file, even if you had executed the design commands using your own Tcl script. The exported Tcl file only contains the tool commands and their accompanying arguments.

extended_run_lib - Libero SoC Only

Note: This is not a Tcl command; it is a shell script that can be run from the command line.

The extended_run_lib Tcl script enables you to run the multiple pass layout in batch mode from a command line.

```
$ACTEL_SW_DIR/bin/libero script:$ACTEL_SW_DIR/scripts/extended_run_lib.tcl
logfile:extended_run.log “script_args:-root path/designer/module_name [-n numPasses] [-
starting_seed_index numIndex] [-compare_criteria value] [-c clockName] [-analysis value] [-
slack_criteria value] [-stop_on_success] [-timing_driven|-standard] [-power_driven value]
[-placer_high_effort value]”
```

Note:

- There is no option to save the design files from all the passes. Only the (Timing or Power) result reports from all the passes are saved.
- This script supports only SmartFusion2, IGLOO2 and RTG4 designs.

Arguments

`-root path/designer/module_name`

The path to the root module located under the designer directory of the Libero project.

`[-n numPasses]`

Sets the number of passes to run. The default number of passes is 5.

`[-starting_seed_index numIndex]`

Indicates the specific index into the array of random seeds which is to be the starting point for the passes. Value may range from 1 to 100. If not specified, the default behavior is to continue from the last seed index that was used.

`[-compare_criteria value]`

Sets the criteria for comparing results between passes. The default value is set to frequency when the –c option is given or timing constraints are absent. Otherwise, the default value is set to violations.

Value	Description
frequency	Use clock frequency as criteria for comparing the results between passes. This option can be used in conjunction with the -c option (described below).
violations	Use timing violations as criteria for comparing the results between passes. This option can be used in conjunction with the -analysis, -slack_criteria and -stop_on_success options (described below).

Value	Description
power	Use total power as criteria for comparing the results between passes, where lowest total power is the goal.

`[-c clockName]`

Applies only when the clock frequency comparison criteria is used. Specifies the particular clock that is to be examined. If no clock is specified, then the slowest clock frequency in the design in a given pass is used. The clock name should match with one of the Clock Domains in the Summary section of the Timing report.

`[-analysis value]`

Applies only when the timing violations comparison criteria is used. Specifies the type of timing violations (the slack) to examine. The following table shows the acceptable values for this argument:

Value	Description
max	Examines timing violations (slack) obtained from maximum delay analysis. This is the default.
min	Examines timing violations (slack) obtained from minimum delay analysis.

`[-slack_criteria value]`

Applies only when the timing violations comparison criteria is used. Specifies how to evaluate the timing violations (slack). The type of timing violations (slack) is determined by the -analysis option. The following table shows the acceptable values for this argument:

Value	Description
worst	Sets the timing violations criteria to Worst slack. For each pass obtains the most amount of negative slack (or least amount of positive slack if all constraints are met) from the timing violations report. The largest value out of all passes will determine the best pass. This is the default.
tns	Sets the timing violations criteria to Total Negative Slack (tns). For each pass it obtains the sum of negative slack values from the first 100 paths from the timing violations report. The largest value out of all passes determines the best pass. If no negative slacks exist for a pass, then the worst slack is used to evaluate that pass.

`[-stop_on_success]`

Applies only when the timing violations comparison criteria is used. The type of timing violations (slack) is determined by the -analysis option. Stops running the remaining passes if all timing constraints have been met (when there are no negative slacks reported in the timing violations report).

`[-timing_driven|-standard]`

Sets layout mode to timing driven or standard (non-timing driven). The default is -timing_driven or the mode used in the previous layout command.

`[-power_driven value]`

Enables or disables power-driven layout. The default is off or the mode used in the previous layout command. The following table shows the acceptable values for this argument:

Value	Description
off	Does not run power-driven layout.

Value	Description
on	Enables power-driven layout.

`[-placer_high_effort value]`

Sets placer effort level. The default is off or the mode used in the previous layout command. The following table shows the acceptable values for this argument:

Value	Description
off	Runs layout in regular effort.
on	Activates high effort layout mode.

Return

A non-zero value will be returned on error.

Supported Families

SmartFusion2, IGLOO2, RTG4

Exceptions

None

Example

```
D:/Libero_11_3_SP1/Designer/bin/libero
script:D:/Libero_11_3_SP1/Designer/scripts/extended_run_lib.tcl logfile:extended_run.log
"script_args:-root E:/designs/centralfpga/designer/centralfpga -n 3 -slack_criteria tns -
stop_on_success"
```

See Also

Place and Route - SmartFusion2, IGLOO2, and RTG4

Multiple Pass Layout

extended_run_shell - Designer Only

Note: This is not a Tcl command; it is a shell script that can be run from the command line. To invoke multiple pass layout within another Designer Tcl script, refer to extended_run_gui.

The extended_run_shell Tcl script enables you to run the multiple pass layout in batch mode from a command line. Use this script from the tcl shell "acttclsh". **This is the script or command-line equivalent to using the multiple pass layout in the GUI.**

```
$ACTEL_SW_DIR/bin/acttclsh extended_run_shell.tcl -adb adbFileName.adb [-n numPasses] [-
starting_seed_index numIndex] [-save_all] [-compare_criteria value] [-c clockName] [-
analysis value] [-slack_criteria value] [-timing_driven|-standard] [-stop_on_success] [-
seq_opt value][-run_placer value] [-place_incremental value] [-route_incremental value] [-
placer_high_effort value] [-mindel_repair value] [-power_driven value]
```

Arguments

`-adb adbFileName.adb`

This is the design file to run multiple passes of layout.

`[-n numPasses]`

Sets the number of passes to run. The default number of passes is 5.

`[-starting_seed_index numIndex]`

Indicates the specific index into the array of random seeds which is to be the starting point for the passes. Its value should range from 1 to 101. If not specified, the default behavior is to continue from the last seed index which was used.

`[-save_all]`

Saves all intermediate designs in<adbFileName>_r<runNum>_s<seedIndex>.adb. The best result is also stored to the original *.adb file as well. The default behavior does not save all results.

`[-compare_criteria value]`

The following table shows the acceptable values for this argument:

Value	Description
frequency	Sets the criteria for comparing results between passes to be clock frequency based. This is the default. This option enables the -c option (described below).
violations	Sets the criteria for comparing results between passes to be timing violations (slack) based. This option enables the -analysis, -slack_criteria, and -stop_on_success options (described below).
power	Sets the criteria for comparing results between passes to be based on the lowest total power.

`[-c clockName]`

Applies only when the clock frequency comparison criteria is used. Specifies the particular clock that is to be examined. If no clock is specified, then the slowest clock frequency in the design in a given pass is used.

`[-analysis value]`

Applies only when the timing violations comparison criteria is used. The following table shows the acceptable values for this argument:

Value	Description
max	Examines timing violations (slacks) obtained from maximum delay analysis. This is the default.
min	Examines timing violations (slacks) obtained from minimum delay analysis.

`[-slack_criteria value]`

Applies only when the timing violations comparison criteria is used. The type of timing violations (slacks) is determined by the -analysis option. The following table shows the acceptable values for this argument:

Value	Description
worst	Sets the timing violations criteria to worst slack. For each pass obtains the most amount of negative slack (or least amount of positive slack if all constraints are met) from the timing violations report. The largest value out of all passes will determine the best pass. This is the default.
tns	Sets the timing violations criteria to total negative slack. For each pass obtains the sum of negative slacks from the first 100 paths from the timing violations report. The largest value out of all passes will determine the best pass. If no negative slacks exist for a pass, then the worst slack is used to evaluate that

Value	Description
	pass

`[-stop_on_success]`

Applies only when the timing violations comparison criteria is used. The type of timing violations (slacks) is determined by the -analysis option. Stops performing remaining passes if all timing constraints have been met (when there are no negative slacks reported in the timing violations report).

`-seq_opt` *`value`*

The following table shows the acceptable values for this argument:

Value	Description
off	Disables physical synthesis of sequential logic. This is the default.
on	Enables physical synthesis of sequential logic in high-effort mode

`[-timing_driven|-standard]`

Sets layout mode to be timing driven or standard (non-timing driven). The default is -timing_driven or the mode used in the previous layout command.

`[-run_placer` *`value`*`]`

The following table shows the acceptable values for this argument:

Value	Description
on	Invokes placer. This is the default.
off	Skips placer.

`[-place_incremental` *`value`*`]`

The following table shows the acceptable values for this argument:

Value	Description
off	Discards previous placement. This is the default.
on	Sets the previous placement as the initial starting point for each pass.
fix	Locks previous placement for each pass.

`[-route_incremental` *`value`*`]`

The following table shows the acceptable values for this argument:

Value	Description
off	Discards previous routing. This is the default.
on	Sets the previous routing as the initial starting point for each pass.

`[-placer_high_effort` *`value`*`]`

This is an advanced option that is available only for SmartFusion, IGLOO, ProASIC3 and Fusion families. The following table shows the acceptable values for this argument:

Value	Description
off	Runs layout in regular effort. This is the default.
on	Activates high effort layout mode.

`[-mindel_repair value]`

This is an advanced option that is available only for SmartFusion, IGLOO, ProASIC3 and Fusion families. The following table shows the acceptable values for this argument:

Value	Description
off	Does not run minimum delay violations repair. This is the default.
on	Enables repair of minimum delay violations during route.

`[-power_driven value]`

This option is available only for SmartFusion, IGLOO, ProASIC3 and Fusion families. The following table shows the acceptable values for this argument:

Value	Description
off	Does not run power-driven layout. This is the default.
on	Enables power-driven layout.

Return

A non-zero value will be returned on error.

Supported Families

SmartFusion, IGLOO, ProASIC3 and Fusion

Exceptions

None

Example

1. On *my.adb*, run 5 (default) passes continuing from the last seed index using slowest clock frequency (default) comparison criteria.
```
% acttclsh extended_run_shell.tcl -adb my.adb
```
2. On *my.adb*, run 3 passes starting with seed index 6, saving all results, using clock frequency comparison criteria for clock "PCI_CLK".
```
% acttclsh extended_run_shell.tcl -adb my.adb -n 3 -starting_seed_index 6 -save_all -c PCI_CLK
```
3. On my.adb, run 5 (default) passes continuing from the last seed index, saving all results, using timing violations comparison criteria with maximum delay (default) analysis and worst slack (default) criteria; invoke high effort layout.
```
% acttclsh extended_run_shell.tcl -adb my.adb -save_all -compare_criteria violations –placer_high_effort on
```
4. On my.adb, run 5 (default) passes continuing from the last seed index, saving all results, using timing violations comparison criteria with maximum delay (default) analysis and total negative slack criteria; invoke placement effort level 5.

```
% acttclsh extended_run_shell.tcl -adb my.adb -save_all -compare_criteria violations
-slack_criteria tns –effort_level 5
```

5. On my.adb, run 5 (default) passes continuing from the last seed index, saving all results, using timing violations comparison criteria with minimum delay analysis and worst slack (default) criteria; stop if there are no violations.
```
% acttclsh extended_run_shell.tcl -adb my.adb -save_all -compare_criteria violations
-analysis min –stop_on_success
```
6. On my.adb, run 5 (default) passes continuing from the last seed index, saving all results, using timing violations comparison criteria with minimum delay analysis and total negative slack criteria; invoke repair of minimum delay violations.
```
% acttclsh extended_run_shell.tcl -adb my.adb -save_all -compare_criteria violations
-analysis min -slack_criteria tns –mindel_repair on
```

See Also

Running Layout

Multiple Pass Layout

extended_run_gui

Sample Tcl Script - Project Manager

The following Tcl commands create a new project named proj1 and sets your project options.

```
#Create new project
new_project -name proj1 -location c:/actelprj -family fusion –die AFS090 -package "108
QFN" -hdl VHDL
#Import HDL source file named hdlsource1.vhd
import_files -hdl_source c:\hdlsource1.vhd
#Run synthesis and create a logfile named synth1.
run_synthesis -logfile synth.log
# he default ADB file, run Compile, run Layout
run_designer -logfile designer_log -adb new -compile TRUE -layout TRUE -export_ba TRUE
```

Tcl Flow in the Libero SoC

Use the following commands to manage and build your project in the Libero SoC.

Design Flow in the Project Manager

The Tcl commands below outline the entire design flow. Once you create a project in the Project Manager you can use the commands below to complete every operation from synthesis to generating an HDL netlist. Click any command to go to the command definition.

run_synthesis [-logfile *name*]

run_simulation [-logfile *name*]

check_hdl -file *filename*

check_schematic -file *filename*

create_symbol [-module *module*]

export_io_constraints_from_adb -adb *filename* -output *outputfilename*

generate_ba_files -adb *filename*

generate_hdl_from_schematic [-module *modulename*]

generate_hdl_netlist [-netlist *filename*] [-run_drc *"TRUE | FALSE"*]

rollback_constraints_from_adb -adb *filename* -output *output_filename*

run_designer [-logfile *filename*] [-script "*script to append*"] [-append_commands "*commands to execute*"] [-adb "*new | open | default*"] [-compile "*TRUE | FALSE*"] [-layout "*TRUE | FALSE*"] [-export_ba "*TRUE | FALSE*"]

run_drc [-netlist *file*] [-gen_hdl *"TRUE | FALSE"*]

Manage Profiles in the Project Manager

add_profile -name *profilename* -type "*synthesis | simulation | stimulus | flashpro | physynth | coreconfig*" -tool *profiletool* -location *tool_location* [-args *tool_parameters*] [-batch "*TRUE | FALSE*"]

edit_profile -name *profilename* -type "*synthesis | simulation | stimulus | flashpro | physynth | coreconfig*" -tool *profiletool* -location *tool_location* [-args *tool_parameters*] [-batch "*TRUE | FALSE*"] [-new_name *name*]

export_profiles -file *name* [-export "*predefined | user | all*"]

remove_profile -name *profile_name*

select_profile -name *profile_name*

Linking Files

change_link_source -file *filename* -path *pathname*

create_links [-hdl_source *file*]* [-stimulus *file*]* [-sdc *file*]* [-pin *file*]* [-dcf *file*]* [-gcf *file*]* [-pdc *file*]* [-crt *file*]* [-vcd *file*]*

export_as_link -file *filename* -path *link_path*

unlink -file *file* [-local *local_filename*]

Set Simulation Options in the Project Manager

add_modelsim_path -lib *library_name* [-path *library_path*] [-remove " "]

Set Device in the Project Manager

set_device [-family *family*] [-die *die*] [-package *package*]

Miscellaneous Operations in the Project Manager

project_settings [-hdl "*VHDL | VERILOG*"] [-auto_update_modelsim_ini "*TRUE | FALSE*"] [-auto_update_viewdraw_ini "*TRUE | FALSE*"] [-block_mode "*TRUE | FALSE*"] [-auto_generate_synth_hdl "*TRUE | FALSE*"] [-auto_run_drc "*TRUE | FALSE*"] [-auto_generate_viewdraw_hdl "*TRUE | FALSE*"] [-auto_file_detection "*TRUE | FALSE*"]

refresh

set_option [-synth "*TRUE | FALSE*"] [-module "*module_name*"]

remove_core -name *core_name*

All Families - Project Manager Tcl Commands

add_file_to_library

Tcl command; adds a file to a library in your project.

```
add_file_to_library
-library name
-file name
```

Arguments

`-library name`

Name of the library where you wish to add your file.

`-file name`

Specifies the new name of the file you wish to add (must be a full pathname).

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Add a file named foo.vhd from the ./project/hdl directory to the library 'my_lib'

```
add_file_to_library -library my_lib -file ./project/hdl/foo.vhd
```

See Also

add_library
remove_library
rename_library
Project Manager Tcl Command Reference

add_library

Tcl command; adds a VHDL library to your project.

```
add_library
-library name
```

Arguments

`-library name`

Specifies the name of your new library.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Create a new library called 'my_lib'.

```
add_library –library my_lib
```

See Also

remove_library

rename_library

Project Manager Tcl Command Reference

add_modelsim_path

Tcl command; adds a ModelSim simulation library to your project.

```
add_modelsim_path -lib library_name [-path library_path] [-remove " "]
```

Arguments

`-lib library_name`

Name of the library you want to add.

`-path library_path`

Path to library that you want to add.

`-remove " "`

Name of library you want to remove (if any).

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Add the ModelSim library 'msim_update2' located in the c:\modelsim\libraries directory and remove the library 'msim_update1':

```
add_modelsim_path -lib msim_update2 [-path c:\modelsim\libraries] [-remove msim_update1]
```

See Also

Project Manager Tcl Command Reference

add_profile

Tcl command; sets the same values as the Add or Edit Profile dialog box.

```
add_profile -name profilename -type value -tool profiletool -location tool_location [-args
tool_parameters] [-batch value]
```

Arguments

`-name profilename`

Specifies the name of your new profile.

`-type value`

Specifies your profile type, where value is one of the following:

Value	Description
synthesis	New profile for a synthesis tool
simulation	New profile for a simulation tool
stimulus	New profile for a stimulus tool

Value	Description
flashpro	New FlashPro tool profile

`-tool profiletool`

Name of the tool you are adding to the profile.

`-location tool_location`

Full pathname to the location of the tool you are adding to the profile.

`-args tool_parameters`

Profile parameters (if any).

`-batch value`

Runs the tool in batch mode (if TRUE). Possible values are:

Value	Description
TRUE	Runs the profile in batch mode
FALSE	Does not run the profile in batch mode

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Create a new FlashPro tool profile called 'myflashpro' linked to a FlashPro installation in my c:\programs\actel\flashpro\bin directory

```
new_profile -name myflashpro -type flashpro -tool flashpro.exe -location
c:\programs\actel\flashpro\bin\flashpro.exe -batch FALSE
```

See Also

Project Manager Tcl Command Reference

associate_stimulus

Tcl command; associates a stimulus file in your project.

```
-associate_stimulus
[-file name]*
[-mode value]
-module value
```

Arguments

`-file name`

Specifies the name of the file to which you want to associate your stimulus files.

`-mode value`

Specifies whether you are creating a new stimulus association, adding, or removing; possible values are:

Value	Description
new	Creates a new stimulus file association

Value	Description
add	Adds a stimulus file to an existing association
remove	Removes an stimulus file association

`-module value`

Sets the module, where value is the name of the module.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The example associates a new stimulus file 'stim.vhd' for stimulus.

```
-associate_stimulus -file stim.vhd -mode new -module stimulus
```

See Also

Project Manager Tcl Command Reference

change_link_source

Tcl command; changes the source of a linked file in your project.

```
change_link_source -file filename -path new_source_path
```

Arguments

`-file filename`

Name of the linked file you want to change.

`-path new_source_path`

Location of the file you want to link to.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Change the link to a file 'sim1.vhd' in your project and link it to the file in c:\microsemi\link_source\simulation_test.vhd

```
change_link_source -file sim1.vhd -path c:\microsemi\link_source\simulation_test.vhd
```

See Also

Project Manager Tcl Command Reference

check_hdl

Tcl command; checks the HDL in the specified file.

```
check_hdl -file filename
```

Arguments

`-file filename`

Name of the HDL file you want to check.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Check HDL on the file hdl1.vhd.

```
check_hdl -file hdl1.vhd
```

See Also

Project Manager Tcl Command Reference

check_schematic

Tcl command; checks the schematic.

```
check_schematic -file filename
```

Arguments

`-file filename`

Name of the schematic file you want to check.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Check schematic on the file schem.2vd.

```
check_schematic -file schem.2vd
```

See Also

Project Manager Tcl Command Reference

close_design (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; closes the current design and brings Designer to a fresh state to work on a new design. This is equivalent to selecting the Close command from the File menu.

```
close_design
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
if  { [catch { close_design }] {
            Puts “Failed to close design”
            # Handle Failure
} else {
            puts “Design closed successfully”
            # Proceed with processing a new design
}
```

See Also

open_design
close_design
new_design
Designer Tcl Command Reference

close_project

Tcl command; closes the current project in Libero SoC. Equivalent to clicking the File menu, and choosing Close Project.

```
close_project
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
close_project
```

See Also

open_project
Project Manager Tcl Command Reference

create_links

Tcl command; creates a link (or links) to a file/files in your project.

```
create_links [-hdl_source file]* [-stimulus file]* [-sdc file]* [-pin file]* [-dcf file]* [-gcf file]* [-pdc file]* [-crt file]* [-vcd file]*
```

Arguments

`-hdl_source file`

Name of the HDL file you want to link.

`-stimulus file`

Name of the stimulus file you want to link.

`-sdc file`

Name of the SDC file you want to link.

`-pin file`

Name of the PIN file you want to link.

`-dcf file`

Name of the DCF file you want to link.

`-gcf file`

Name of the GCF file you want to link.

`-pdc file`

Name of the PDC file you want to link.

`-crt file`

Name of the crt file you want to link.

`-vcd file`

Name of the VCD file you want to link.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Create a link to the file hdl1.vhd.

```
create links [-hdl_source hdl1.vhd]
```

See Also

Project Manager Tcl Command Reference

create_symbol

Tcl command; creates a symbol in a module.

```
create_symbol [-module module]
```

Arguments

`-module module`

Name of the symbol module you want to create.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Create a symbol named mod2.

```
create_symbol [-module mod2]
```

See Also

Project Manager Tcl Command Reference

defvar_get

Tcl command; provides access to the internal variables within Libero and returns its value. This command also prints the value of the variable on the Log window.

```
defvar_get -name variable
```

Arguments

variable

The internal variable.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Example 1: Prints the design name on the log window.

```
    defvar_get -name “DESIGN”
set variableToGet "DESIGN"
set valueOfVariable [defvar_get $variableToGet]
puts "The value is $valueOfVariable"
```

See Also

Project Manager Tcl Command Reference

defvar_set

defvar_set

Tcl command; the defvar_set command sets an internal variable in the Libero system. You must specify at least one argument for this command.

```
defvar_set -name variable -value  value
```

Arguments

Variable must be a valid internal variable and could be accompanied by an optional value. If the *value* is provided, the *variable* is set to the value. If the *value* is null the *variable* is reset.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Example 1:

```
defvar_set -name “FORMAT” -value “VHDL”
```

Sets the FORMAT internal variable to VHDL.

Example 2:

```
set variableToSet "DESIGN"
set valueOfVariable “VHDL”
defvar_set $variableToSet $valueOfVariable
```

These commands set the FORMAT variable to VHDL, shows the use of variables for this command.

See Also

Project Manager Tcl Command Reference

defvar_get

delete_files

Tcl command; deletes files in your Libero SoC project.

```
delete_files
-file value
-from_disk
```

Arguments

`-file value`

Specifies the file you wish to delete from the project. This parameter is required for this Tcl command. It does not delete the file from the disk. Use the -from_disk flag to delete a file from the disk. Value is the name of the file you wish to delete (including the full pathname).

`-from_disk`

Deletes a file from the disk.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Delete the files file1.vhd and file2.vhd from the project, and delete the file top_palace.sdc from the disk.

```
delete_files –file ./project/hdl/file1.vhd –file ./project/hdl/file2.vhd
delete_files –from_disk –file ./project/phy_synthesis/top_palace.sdc
```

The following command deletes the core 'add1' from your disk and project (it is the same as the command to delete an IP core from your disk and project).

```
delete_files -from_disk -file ./project/component/work/add1/add1.cxf
```

See Also

Project Manager Tcl Command Reference

`close_project`

`new_project`

download_core

Tcl command; downloads a core and adds it to your repository.

```
download_core [-vlnv "vlnv"]+ [-location "location"]
```

Arguments

`-vlnv vlnv`

Vendor, library, name and version of the core you want to download.

`-location core_name`

Location of the repository where you wish to add the core.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Download the core CoreAXI to the repository www.actel-ip.com/repositories/SgCore:

```
download_core [-vlnv Actel.DirectCore.COREAXI.2.0.103] [-location www.actel-
ip.com/repositories/SgCore]
```

See Also

Project Manager Tcl Command Reference

edit_profile

Tcl command; sets the same values as the Add or Edit Profile dialog box.

```
edit_profile -name profilename -type value -tool profiletool -location profilelocation [-args parameters] [-batch value] [-new_name name]
```

Arguments

`-name profilename`

Specifies the name of your new profile.

`-type value`

Specifies your profile type, where value is one of the following:

Value	Description
synthesis	New profile for a synthesis tool
simulation	New profile for a simulation tool
stimulus	New profile for a stimulus tool
flashpro	New FlashPro tool profile

`-tool profiletool`

Name of the tool you are adding to the profile.

`-location profilelocation`

Full pathname to the location of the tool you are adding to the profile.

`-args parameters`

Profile tool parameters (if any).

`-batch value`

Runs the tool in batch mode (if TRUE). Possible values are:

Value	Description
TRUE	Runs the profile in batch mode
FALSE	Does not run the profile in batch mode

`-new_name name`

Name of new profile.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Edit a FlashPro tool profile called 'myflashpro' linked to a new FlashPro installation in my c:\programs\actel\flashpro\bin directory, change the name to updated_flashpro.

```
edit_profile -name myflashpro -type flashpro -tool flashpro.exe -location
c:\programs\actel\flashpro\bin\flashpro.exe -batch FALSE -new_name updated_flashpro
```

See Also

Project Manager Tcl Command Reference

export_as_link

Tcl command; exports a file to another directory and links to the file.

```
export_as_link -file filename -path link_path
```

Arguments

`-file filename`

Name of the file you want to export as a link.

`-path link_path`

Path of the link.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Export the file hdl1.vhd as a link to c:\microsemi\link_source.

```
export_as_link -file hdl1.vhd -path c:\microsemi\link_source
```

See Also

Project Manager Tcl Command Reference

export_design_summary

This Tcl command exports an HTML file containing information about your root SmartDesign in your project. The HTML report provides information on:

- Generated Files
- I/Os
- Hardware Instances
- Firmware
- Memory Map

```
export_design_summary –file {D: /Designs/test/sd1.html}
```

Returns

Returns 0 on success, 1 on failure.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

.

export_profiles

Tcl command; exports your tool profiles. Performs the same action as the Export Profiles dialog box.

```
export_profile -file name [-export value]
```

Arguments

`-file name`

Specifies the name of your exported profile.

`-export value`

Specifies your profile export options. The following table shows the acceptable values for this argument:

Value	Description
predefined	Exports only predefined profiles
user	Exports only user profiles
all	Exports all profiles

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following command exports all profiles to the file 'all_profiles':

```
export_profiles -file all_profiles [-export all]
```

See Also

Project Manager Tcl Command Reference

export_script

Tcl command; export_script is a command that explicitly exports the Tcl command equivalents of the current Libero session. You must supply a file name with the -file parameter. You may supply the optional -relative_path parameter to specify whether an absolute or relative path is used in the exported script file.

```
export_script\
-file {<absolute or relative path to constraint file>} \
-relative_path <value> \
```

Arguments

`-file {<absolute or relative path to constraint file>}`

Specifies the absolute or relative path to the constraint file; there may be multiple -file arguments (see example below).

`-relative_path {<value>}`

Sets your option to use a relative or absolute path in the exported script; use 1 for relative path, 0 for absolute.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
export_script -file {./exported.tcl} -relative_path 1
```

generate_hdl_from_schematic

Tcl command; generates an HDL file from your schematic.

```
generate_hdl_from_schematic [-module modulename]
```

Arguments

`-module modulename`

Specifies the module name for your new HDL module

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following example generates a new HDL module module1.vhd:

```
generate_hdl_from_schematic [-module module1.vhd]
```

See Also

Project Manager Tcl Command Reference

import_files (Libero SoC)

Tcl command; enables you to import design source files and constraint files.

SmartFusion2 and IGLOO2 only: For importing constraint files, import_files has retired the -pdc parameter for SmartFusion2 and IGLOO2. It has been replaced with two new parameters to match the new design flow. Physical Design Constraints (PDC) Tcl must now be divided between I/O attribute and pin information from all floorplanning and timing constraints. These commands must now reside in and be imported as separate files. The new parameters specify the type of *.pdc file being imported.

Use of the -pdc parameter with Smartfusion2 or IGLOO2 families will cause an error. The path to the file can be absolute or relative but must be enclosed in curly braces { }.

Use the -can_convert_EDN_to_HDL parameter to convert the EDIF file to HDL and then import the converted HDL file.

Note: The EDIF File is not imported.

```
import_files
-schematic {file}
-symbol {file}
-smartgen_core {file}
-ccp {file}
-stimulus {file}
-hdl_source {file}
-io_pdc {<absolute or relative path to file>} # For PDC containing I/O attribute and pin info
-fp_pdc {<absolute or relative path to file>} # For PDC containing timing and placement info
-edif {file}
-sdc {file}
-pin {file}
-dcf {file}
-pdc {file}
-gcf {file}
```

```
-vcd {file}
-saif {file}
-crt {file}
-simulation {file}
-profiles {file}
-cxf {file}
-templates {file}
-ccz {file}
-wf_stimulus {file}
-modelsim_ini {file}
-can_convert_EDN_to_HDL {true | false
```

Arguments

`-schematic {file}`

Specifies the schematics you wish to import into your IDE project. Type parameter must be repeated for each file.

`-symbol {file}`

Specifies the symbols you wish to import into your IDE project. Type parameter must be repeated for each file.

`-smartgen_core {file}`

Specifies the cores you wish to import into your project. Type parameter must be repeated for each file.

`-ccp {file}`

Specifies the ARM or Cortex-M1 cores you wish to import into your project. Type parameter must be repeated for each file.

`-stimulus {file}`

Specifies HDL stimulus files you wish to import into your project. Type parameter must be repeated for each file.

`-hdl_source {file}`

Specifies the HDL source files you wish to import into your project. Type parameter must be repeated for each file.

`-io_pdc {<absolute or relative path to file>}`

SmartFusion2 and IGLOO2 only - Specifies the PDC file that contains the I/O attribute and pin information.

`-fp_pdc {<absolute or relative path to file>}`

SmartFusion2 and IGLOO2 only - Specifies the PDC file that contains the timing and placement information.

`-edif {file}`

Specifies the EDIF files you wish to import into your project. Type parameter must be repeated for each file. This is a mandatory option if you want to convert EDIF to HDL with the –can_convert_EDN_to_HDL option.

`-can_convert_EDN_to_HDL {true |false |1 | 0} #Boolean {true | false | 1 | 0}`

The –edif option is mandatory. If the –edif option is not specified or the –can_convert_EDN_to_HDL is used with another option, EDIF to HDL conversion will fail.

`-constraint_sdc {file}`

Specifies the SDC constraint files you wish to import into your project. Type parameter must be repeated for each file.

`-constraint_pin {file}`

Specifies the PIN constraint files you wish to import into your project. Type parameter must be repeated for each file.

`-constraint_dcf {file}`

Specifies the DCF constraint files you wish to import into your project. Type parameter must be repeated for each file.

`-constraint_pdc {file}`

Specifies the PDC constraint files you wish to import into your project. Type parameter must be repeated for each file.

`-constraint_gcf {file}`

Specifies the GCF constraint files you wish to import into your project. Type parameter must be repeated for each file.

`-constraint_vcd {file}`

Specifies the VCD constraint files you wish to import into your project. Type parameter must be repeated for each file.

`-constraint_saif {file}`

Specifies the SAIF constraint files you wish to import into your project. Type parameter must be repeated for each file.

`-constraint_crt {file}`

Specifies the CRT constraint files you wish to import into your project. Type parameter must be repeated for each file.

`-simulation {file}`

Specifies the simulation files you wish to import into your Libero SoC project. Type parameter must be repeated for each file.

`-profiles {file}`

Specifies the profile files you wish to import into your Libero SoC project. Type parameter must be repeated for each file.

`-cxf {file}`

Specifies the CXF file (such as SmartDesign components) you wish to import into your Libero SoC project. Type parameter must be repeated for each file.

`-templates {file}`

Specifies the template file you wish to import into your IDE project.

`-ccz {file}`

Specifies the IP core file you wish to import into your project.

`-wf_stimulus {file}`

Specifies the WaveFormer Pro stimulus file you wish to import into your project.

`-modelsim_ini {file}`

Specifies the ModelSIM INI file that you wish to import into your project.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The command below imports the HDL source files file1.vhd and file2.vhd:

```
import_files -hdl_source file1.vhd –hdl_source file2.vhd
```

See Also

Project Manager Tcl Command Reference

new_project

Tcl command; creates a new project in the Libero SoC. If you do not specify a location, Libero SoC saves the new project in your current working directory.

```
new_project -name project_name -location project_location -family family_name –die device_die -
package package_name -hdl HDL_type -speed speed_grade -die_voltage value -
standalone_peripheral_initialization {1 | 0} -adv_options value
```

Arguments

`-name` *`project_name`*

The name of the project. This is used as the base name for most of the files generated from Libero SoC.

`-location` *`project_location`*

The location of the project. Must not be an existing directory.

`-family` *`family_name`*

The Microsemi SoC device family for your targeted design.

`-die` *`device_die`*

Die for your targeted design.

`-package` *`package_name`*

Package for your targeted design.

`-hdl` *`HDL_type`*

Sets the HDL type for your new project.

Value	Description
VHDL	Sets your new projects HDL type to VHDL
VERILOG	Sets your new projects to Verilog

`-speed` *`speed_grade`*

Sets the speed grade for your project. Possible values depend on your device, die and package. See your device datasheet for details.

`-die_voltage` *`value`*

Sets the die voltage for your project. Possible values depend on your device. See your device datasheet for details.

`-standalone_peripheral_initialization {1 | 0}` *(for SmartFusion2 and IGLOO2 only)*

When set to 1, this option instructs System Builder not to build the initialization circuitry for your Peripherals. Set this option to 1 if you want to build your own peripheral initialization logic in SmartDesign to initialize each of the peripherals (MDDR/FDDR/SERDES) independently.

`-adv_options` *`value`*

Sets your advanced options, such as operating conditions.

Value	Description
IO_DEFT_STD:LVTTL	Sets your I/O default value to LVTTL
TEMPR:MIL	Sets your default temperature range for operating condition analysis; can be COM (Commercial), MIL (Military) or IND (Industrial).
VCCI_1.5_VOLTR:COM	Sets VCCI to 1.5 and voltage range to Commercial
VCCI_1.8_VOLTR:COM	Sets VCCI to 1.8 and voltage range to Commercial
VCCI_2.5_VOLTR:COM	Sets VCCI to 2.5 and voltage range to Commercial
VCCI_3.3_VOLTR:COM	Sets VCCI to 3.3 and voltage range to Commercial

Value	Description
VOLTR:COM	Sets your voltage range for operating condition analysis; can be COM (Commercial), MIL (Military) or IND (Industrial).
PART_RANGE:MIL	Sets your default temperature range for your project; can be COM (Commercial), MIL (Military) or IND (Industrial).

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Creates a new project in the directory ./designs/mydesign, with the HDL type Verilog for the SmartFusion2 family.

```
new_project -location {./designs/mydesign} -name {mydesign}
-standalone_peripheral_initialization 1 -hdl {VERILOG} -family
{SmartFusion2} -die {M2S150TS} -package {FCS536} -speed {-1} -die_voltage {1.2}
-adv_options {DSW_VCCA_VOLTAGE_RAMP_RATE:100_MS} -adv_options
{IO_DEFT_STD:LVCMOS 2.5V} -adv_options {PLL_SUPPLY:PLL_SUPPLY_25} -adv_options
{RESTRICTPROBEPINS:1} -adv_options {SYSTEM_CONTROLLER_SUSPEND_MODE:0}
-adv_options {TEMPR:IND} -adv_options {VCCI_1.2_VOLTR:IND} -adv_options
{VCCI_1.5_VOLTR:IND} -adv_options {VCCI_1.8_VOLTR:IND} -adv_options
{VCCI_2.5_VOLTR:IND} -adv_options {VCCI_3.3_VOLTR:IND} -adv_options {VOLTR:IND}
```

See Also

Project Manager Tcl Command Reference

open_project

Tcl command; opens an existing Libero SoC project.

```
open_project project_name-do_backup_on_convert value-backup_file backup_filename
```

Arguments

project_name

Must include the complete path to the PRJ file. If you do not provide the full path, Libero SoC infers that you want to open the project from your current working directory.

`-do_backup_on_convert` *value*

Sets the option to backup your files if you open a project created in a previous version of Libero SoC.

Value	Description
TRUE	Creates a backup of your original project before opening
FALSE	Opens your project without creating a backup

`-backup_file` *backup_filename*

Sets the name of your backup file (if you choose to do_backup_on_convert).

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Open project.prj from the c:/netlists/test directory.

```
open_project c:/netlists/test/project.prj
```

See Also

close_project
new_project
save_project
Project Manager Tcl Command Reference

organize_cdbs

Tcl command; enables you to organize the CDB files in your project.

```
organize_cdbs -file name -module name
```

Arguments

`-file` *name*

Specifies the name of the CDB file you intend to organize.

`-module` *name*

Identifies the name of the module to which you wish to add the CDB file.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Adds the file design2.cdb to the module design_test.

```
organize_cdbs -file design2.cdb -module design_test
```

See Also

Project Manager Tcl Command Reference

organize_constraints

Tcl command; organizes the constraint files in your project.

```
-organize_constraints
[-file name]*
[-mode value]
-designer_view name
-module value
-tool value
```

Arguments

`-file` *name*

Specifies the name of the file to which you want to associate your stimulus files.

`-mode` *value*

Specifies whether you are creating a new stimulus association, adding, or removing; possible values are:

Value	Description
new	Creates a new stimulus file association
add	Adds a stimulus file to an existing association
remove	Removes an stimulus file association

`-designer_view name`

Sets the name of the Designer View in which you wish to add the constraint file, where name is the name of the view (such as impl1).

`-module value`

Sets the module, where value is the name of the module.

`-tool value`

Identifies the intended use for the file, possible values are:

Value	Description
synthesis	File to be used for synthesis
designer	File to be used in Designer
phsynth	File to be used in physical synthesis

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The example adds the constraint file delta.vhd in the Designer View impl2 for the Designer tool.

```
-organize_constraints -file delta.vhd -mode new -designer_view impl2 -module constraint
-tool designer
```

See Also

Project Manager Tcl Command Reference

organize_sources

Tcl command; organizes the source files in your project.

Arguments

```
-organize_sources
[-file name]*
[-mode value]
-module value
-tool value
[-use_default value]
```

Arguments

`-file` *name*

Specifies the name of the file to which you want to associate your stimulus files.

`-mode` *value*

Specifies whether you are creating a new stimulus association, adding, or removing; possible values are:

Value	Description
new	Creates a new stimulus file association
add	Adds a stimulus file to an existing association
remove	Removes an stimulus file association

`-module` *value*

Sets the module, where value is the name of the module.

`-tool` *value*

Identifies the intended use for the file, possible values are:

Value	Description
synthesis	File to be used for synthesis
simulation	File to be used for simulation

`-use_default` *value*

Uses the default values for synthesis or simulation; possible values are:

Value	Description
TRUE	Uses default values for synthesis or simulation.
FALSE	Uses user-defined values for synthesis or simulation

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The example organizes a new stimulus file 'stim.vhd' using default settings.

```
-organize_sources -file stim.vhd -mode new -module stimulus -tool synthesis -use_default
TRUE
```

See Also

Project Manager Tcl Command Reference

project_settings

Tcl command; modifies project flow settings for your Libero SoC project.

```
project_settings [-hdl "VHDL | VERILOG"] [-auto_update_modelsim_ini "TRUE | FALSE"] [-
auto_update_viewdraw_ini "TRUE | FALSE"] [-block_mode "TRUE | FALSE"] [-vm_netlist_flow TRUE
```

```
| FALSE | 1 | 0][-auto_generate_synth_hdl "TRUE | FALSE"] [-auto_run_drc "TRUE | FALSE"] [-
auto_generate_viewdraw_hdl "TRUE | FALSE"] [-auto_file_detection "TRUE | FALSE"] [-
standalone_peripheral_initialization "1 | 0"]
```

Arguments

`-hdl "VHDL | VERILOG"`

Sets your project HDL type.

`-auto_update_modelsim_ini "TRUE | FALSE"`

Sets your auto-update modelsim.ini file option. TRUE updates the file automatically.

`-auto_update_viewdraw_ini "TRUE | FALSE"`

Sets your auto-update viewdraw.ini file option. TRUE updates the file automatically.

`-block_mode "TRUE | FALSE"`

Puts the Project Manager in Block mode, enables you to create blocks in your project.

`-vm_netlist_flow "TRUE | FALSE | 1 | 0" (SmartFusion 2 and IGLOO 2 only)`

Sets to TRUE to generate Verilog netlist from Synthesis. Default is FALSE.

`-auto_generate_synth_hdl "TRUE | FALSE"`

Auto-generates your HDL file after synthesis (when set to TRUE).

`-auto_run_drc "TRUE | FALSE"`

Auto-runs the design rule check immediately after synthesis (when set to TRUE).

`-auto_generate_viewdraw_hdl "TRUE | FALSE"`

Auto-generates your HDL netlist after a Save & Check in ViewDraw (when set to TRUE).

`-auto_file_detection "TRUE | FALSE"`

Automatically detects when new files have been added to the Libero SoC project folder (when set to TRUE).

`-standalone_peripheral_initialization "1|0"`

When set to 1, this option instructs System Builder not to build the initialization circuitry for your Peripherals. Set this option to 1 if you want to build your own peripheral initialization logic in SmartDesign to initialize each of the peripherals (MDDR/FDDR/SERDES) independently.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Set your project to VHDL, do not auto-update the ModelSim INI or ViewDraw INI files, auto-generate HDL after synthesis, and enable auto-detect for files.

```
project_settings [-hdl "VHDL"] [-auto_update_modelsim_ini "FALSE"] [-
auto_update_viewdraw_ini "FALSE"] [-block_mode "FALSE"] [-auto_generate_synth_hdl
"TRUE"] [-auto_file_detection "TRUE"]
```

See Also

Project Manager Tcl Command Reference

refresh

Tcl command; refreshes your project, updates the view and checks for updated links and files.

```
refresh .
```

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
refresh
```

See Also

Project Manager Tcl Command Reference

remove_core

Tcl command; removes a core from your project.

```
remove_core -name core_name
```

Arguments

`-name core_name`

Name of the core you want to remove.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Remove the core ip-beta2:

```
remove_core -name ip-beta2.ccz
```

See Also

Project Manager Tcl Command Reference

remove_library

Tcl command; removes a VHDL library from your project.

```
remove_library
-library name
```

Arguments

`-library name`

Specifies the name of the library you wish to remove.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Remove (delete) a library called 'my_lib'.

```
remove_library –library my_lib
```

See Also

Project Manager Tcl Command Reference

`add_library`

`rename_library`

remove_profile

Tcl command; deletes a tool profile.

```
remove_profile -name profilename
```

Arguments

`-name profilename`

Specifies the name of the profile you wish to delete.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following command deletes the profile 'custom1':

```
remove_profile -name custom1
```

See Also

Project Manager Tcl Command Reference

rename_library

Tcl command; renames a VHDL library in your project.

```
rename_library
-library name
 -name name
```

Arguments

`-library name`

Identifies the current name of the library that you wish to rename.

`-name name`

Specifies the new name of the library.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Rename a library from 'my_lib' to 'test_lib1'

```
rename_library –library my_lib -name test_lib1
```

See Also

Project Manager Tcl Command Reference

`add_library`

`remove_library`

run_simulation

Tcl command; runs simulation on your project with your default simulation tool and creates a logfile.

```
run_simulation [-logfile "name"] [-wlf "name"] [-dofile "name"] [-refresh_lib "value"] [-
state "value"]
```

Arguments

`-logfile "name"`

Name of your simulation logfile.

`-wlf "name"`

Name of WLF file you wish to use; this command and the -dofile command are exclusive.

`-dofile "name"`

Name of DO file you wish to use; this command and the -wlf command are exclusive.

`-refresh_lib "value"`

Sets your library refresh option using one of the following values:

Value	Description
TRUE	Refreshes your simulation library
FALSE	Does not refresh your simulation library

`-state "value"`

Identifies which simulation you want to perform.

Value	Description
Pre_Synthesis	Runs pre-synthesis simulation
Post_Synthesis	Runs post-synthesis simulation
Post_Phy_Synthesis	Runs post-synthesis physical simulation
Post_Layout	Runs post-layout simulation

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following command runs post-layout simulation on your project using the DO file 'myfile.do', does not refresh the simulation library, and creates the logfile 'mylog.log':

```
run_simulation -logfile “Mylog.log” -dofile “Myfile.do” -refresh_lib "TRUE" -state
“Post_Layout”
```

See Also

Project Manager Tcl Command Reference

`run_synthesis`

run_synthesis

Tcl command; runs synthesis on your project and creates a logfile.

```
run_synthesis [-logfile "name"] [-target "target file name"]
```

Arguments

`-logfile "name"`

Name of your synthesis logfile.

`-target "target file name"`

Name of your synthesis target file.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Run synthesis on your project and create the logfile 'mysynlogfile', and creates the target file 'targfile'.

```
run_synthesis [-logfile "mysynlogfile"] [-target "targfile"]
```

See Also

Project Manager Tcl Command Reference

`run_simulation`

save_project_as

Tcl command; the save_project_as command saves the current project in Libero SoC with a different name and in a specified directory. You must specify a location with the -location parameter.

```
save_project_as
-name project_name
-location project_location
-files value
-designer_views value
-replace_links value
```

Arguments

`-name project_name`

Specifies the name of your new project.

`-location project_location`

Must include the complete path of the PRJ file. If you do not provide the full path, Libero SoC infers that you want to save the project to your current working directory. This is a required parameter.

`-files value`

Specifies the files you want to copy into your new project.

Value	Description
all	Copies all your files into your new project
project	Copies only your Libero SoC project files into your new project
source	Copies only the source files into your new project
none	Copies none of the files into your new project; useful if you wish to manually copy only specific project files

`-designer_views value`

Specifies the Designer views you wish to copy into your new project.

Value	Description
all	Copies all your Designer views into your new project
current	Copies only your current Designer fiew files into your new project
none	Copies none of your views into your new project

`-replace_links` *value*

Specifies whether or not you want to update your file links in your new project.

Value	Description
true	Replaces (updates) the file links in your project during your save
false	Saves your project without updating the file links

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Saves your current Libero SoC project as mydesign.prj in the c:/netlists/testprj/mydesign directory:

```
save_project_as -location c:/netlists/testprj/mydesign -name mydesign.prj
```

See Also

new_project
open_project
save_project
Project Manager Tcl Command Reference

save_design

Tcl command; the save_design command saves the current design in Designer to a file. If filename is not a complete path name, the ADB file is written into the current working directory.

```
save_design filename
```

Arguments

The design is written to a file denoted by the variable filename as an ADB file.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Example 1: Saves the design to a file “test.adb” in the current folder.

```
save_design {test.adb}
```

Example 2: Save design and check if it saved successfully.

```
set designFile {d:/test/my_design.adb}
```

```
if  { [catch { save_design $designFile }] {
              Puts “Failed to save design”
              # Handle Failure
} else {
              puts “Design saved successfully”
              # Proceed to make further changes
}
```

See Also

close_design
new_design
open_design
Designer Tcl Command Reference

save_log

Tcl command; saves your Libero SoC log file.

```
save_log -file value
```

Arguments

`-file value`

Value is your name for the new log file.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Save the log file file_log.

```
save_log -file file_log
```

See Also

close_project
new_project
Project Manager Tcl Command Reference

save_project

Tcl command; the save_project command saves the current project in Libero SoC.

```
save_project
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Saves the project in your current working directory:

```
save_project
```

See Also

new_project
open_project
Project Manager Tcl Command Reference

select_profile

Tcl command; selects a profile to use in your project.

```
select_profile -name profilename
```

Arguments

`-name profilename`

Specifies the name of the profile you wish to use.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following command selects the profile 'custom1':

```
select_profile -name custom1
```

See Also

Project Manager Tcl Command Reference

set_actel_lib_options

Tcl command; the set_actel_lib_options command sets your simulation library to default, or to another library (when you specify a path.

```
set_actel_lib_options -use_default_sim_path value -sim_path {path}
```

Arguments

`-use_default_sim_path value`

Possible values are:

Value	Description
TRUE	Uses the default simulation library.
FALSE	Disables the default simulation library; enables you to specify a different simulation library with the -sim_path {path} option.

`-sim_path {path}`

Specifies the path to your simulation library.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Uses a simulation library in the directory c:\sim_lib\test.

```
set_actel_lib_options -use_default_sim_path FALSE -sim_path {c:\sim_lib\test}
```

See Also

Project Manager Tcl Command Reference

set_device (Project Manager)

Tcl command; sets your device family, die, and package in the Project Manager.

```
set_device [-family family] [-die die] [-package package].[-speed speed_grade] [-adv_options value]
```

Arguments

`-family` *family*

Sets device family.

`-die` *die*

Sets device die.

`-package` *package*

Sets device package.

`-speed` *speed_grade*

Sets device speed grade.

`-adv_options` *value*

Sets your advanced options, such as temperature and voltage settings.

Value	Description
IO_DEFT_STD:LVTTL	Sets your I/O default value to LVTTL
TEMPR:COM	Sets your default temperature range; can be COM (Commercial), MIL (Military) or IND (industrial).
VCCI_1.5_VOLTR:COM	Sets VCCI to 1.5 and voltage range to Commercial
VCCI_1.8_VOLTR:COM	Sets VCCI to 1.8 and voltage range to Commercial
VCCI_2.5_VOLTR:COM	Sets VCCI to 2.5 and voltage range to Commercial
VCCI_3.3_VOLTR:COM	Sets VCCI to 3.3 and voltage range to Commercial
VOLTR:COM	Sets your voltage range; can be COM (Commercial), MIL (Military) or IND (industrial).
RESTRICTPROBEPINS:1	(For SmartFusion2, IGLOO2 and RTG4 only) Sets to 1 to reserve your pins for probing if you intend to debug using SmartDebug.
RESTRICTSPIPINS:1	(RTG4 only) Sets to 1 to reserve pins for SPI functionality in Programming. This reserved SPI pin option is displayed

Value	Description
	in the Compile Report when the compile process completes.
RAD_EXPOSURE:100	(RTG4 only) Specifies the radiation exposure in Krad. Valid range is 0 to 300.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Set your device to Fusion, your die to AFS600, and your package to 484 FBGA

```
set_device [-family fusion] [-die afs600] [-package "484 FBGA"]
```

See Also

Project Manager Tcl Command Reference

set_modelsim_options

Tcl command; sets your Model*Sim* simulation options.

```
set_modelsim_options
[-use_automatic_do_file value]
[-user_do_file {path}]
[-sim_runtime {value}]
[-tb_module_name {value}]
[-tb_top_level_name {value}]
[-include_do_file value
[-included_do_file {value}]
[-type {value}]
[-resolution {value}]
[-add_vsim_options {value}]
[-display_dut_wave value]
[-log_all_signals value]
[-do_file_args value]
[-dump_vcd "TRUE | FALSE"]
[-vcd_file "VCD file name"]
```

Arguments

`-use_automatic_do_file value`

Uses an automatic.do file in your project. Possible values are:

Value	Description
TRUE	Uses the default automatic.do file in your project.
FALSE	Uses a different *.do file; use the other simulation options to specify it.

`-user_do_file {path}`

Specifies the location of your user-defined *.do file.

`-sim_runtime {value}`

Sets your simulation runtime. Value is the number and unit of time, such as {1000ns}.

`-tb_module_name {value}`

Specifies your testbench module name, where value is the name.

`-tb_top_level_name {value}`

Sets the top-level instance name in the testbench, where value is the name.

`-include_do_file value`

Includes a *.do file; possible values are:

Value	Description
TRUE	Includes the *.do file.
FALSE	Does not include the *.do file

`-included_do_file {value}`

Specifies the name of the included *.do file, where value is the name of the file.

`-type {value}`

Resolution type; possible values are:

Value	Description
min	Minimum
typ	Typical
max	Maximum

-resolution {value}

Sets your resolution value, such as {1ps}.

`-add_vsim_options {value}`

Adds more Vsim options, where value specifies the option(s).

-display_dut_wave value

Enables ModelSim to display signals for the tested design; possible values are:

Value	Description
0	Displays the signal for the top_level_testbench
1	Enables ModelSim to display the signals for the tested design

-log_all_signals value

Enables you to log all your signals during simulation; possible values are:

Value	Description
TRUE	Logs all signals
FALSE	Does not log all signals

`-do_file_args value`

Specifies *.do file command parameters.

`-dump_vcd value`

Dumps the VCD file when simulation is complete; possible values are:

Value	Description
TRUE	Dumps the VCD file
FALSE	Does not dump the VCD file

`-vcd_file {value}`

Specifies the name of the dumped VCD file, where value is the name of the file.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Sets ModelSim options to use the automatic *.do file, sets simulation runtime to 1000ns, sets the testbench module name to "testbench", sets the testbench top level to <top>_0, sets simulation type to "max", resolution to 1ps, adds no vsim options, does not log signals, adds no additional DO file arguments, dumps the VCD file with a name power.vcd.

```
set_modelsim_options -use_automatic_do_file 1 -sim_runtime {1000ns} -tb_module_name {testbench} -tb_top_level_name {<top>_0} -include_do_file 0 -type {max} -resolution {1ps} -add_vsim_options {} -display_dut_wave 0 -log_all_signals 0 -do_file_args {} -dump_vcd 0 -vcd_file {power.vcd}
```

See Also

Project Manager Tcl Command Reference

set_option

Tcl command; sets your synthesis options on a module.

```
set_option [-synth "TRUE | FALSE"] [-module "module_name"]
```

Arguments

`-synth "TRUE | FALSE"`

Runs synthesis (for a value of TRUE).

`-module module_name`

Identifies the module on which you will run synthesis.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Run synthesis on the module test1.vhd:

```
set_option [-synth TRUE] [-module <module_name>]
```

See Also

Project Manager Tcl Command Reference

set_user_lib_options

Tcl command; sets your user library options during simulation. If you do not use a custom library these options are not available.

```
set_user_lib_options
-name {value}
-path {path}
-option {value}
```

Arguments

`-name {value}`

Sets the name of your user library.

`-path {path}`

Sets the pathname of your user library.

`-option {value}`

Sets your default compile options on your user library; possible values are:

Value	Description
do_not_compile	User library is not compiled
refresh	User library is refreshed
compile	User library is compiled
recompile	User library is recompiled
refresh_and_compile	User library is refreshed and compiled

Supported Familes

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The example below sets the name for the user library to "test1", the path to c:/actel_des_files/libraries/test1, and the compile option to "do not compile".

```
set_user_lib_options -name {test1} -path {c:/actel_des_files/libraries/test1} -option
{do_not_compile}
```

See Also

Project Manager Tcl Command Reference

set_root

Tcl command; sets the module you specify as the root.

```
set_root module_name
```

Arguments

`set_root module_name`

Specifies the name the module you want to set as root.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Set the module mux8 as root:

```
set_root mux8
```

See Also

Project Manager Tcl Command Reference

unlink

Tcl command; removes a link to a file in your project.

```
unlink -file filename [-local local_filename]
```

Arguments

`-file filename`

Name of the linked (remote) file you want to unlink.

`-local local_filename`

Name of the local file that you want to unlink.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Unlink the file hdl1.vhd from my local file test.vhd

```
unlink -file hdl1.vhd [-local test.vhd]
```

See Also

Project Manager Tcl Command Reference

use_file

Tcl command; specifies which file in your project to use.

```
use_file
-file value
 -module value
 -designer_view value
```

Arguments

`-file`*value*

Specifies the EDIF or ADB file you wish to use in the project. Value is the name of the file you wish use (including the full pathname).

`-module` *value*

Specifies the module in which you want to use the file.

`-designer_view` *value*

Specifies the Designer View in which you wish to use the file.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Specify file1.edn in the ./project/synthesis directory, in the module named top, in the Designer View named impl1.

```
use_file –file “./project/synthesis/file1.edn” –module “top” –designer_view “Impl1”
```

See Also

`use_source_file`

Project Manager Tcl Command Reference

use_source_file

Tcl command; defines a module for your project.

```
use_source_file
-file value
 -module value
```

Arguments

`-file` *value*

Specifies the Verilog or VHDL file. Value is the name of the file you wish use (including the full pathname).

`-module` *value*

Specifies the module in which you want to use the file.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Specify file1.vhd in the ./project/hdl directory, in the module named top.

```
use_source_file –file “./project/hdl/file1.vhd” –module “top"
```

See Also

`use_file`

Project Manager Tcl Command Reference

All Families - SmartPower

smartpower_add_new_scenario

Tcl command; creates a new scenario.

```
smartpower_add_new_scenario -name {value} -description {value} -mode {value}
```

Arguments

`-name {value}`

Specifies the name of the new scenario.

`-description {value}`

Specifies the description of the new scenario.

`-mode {<operating mode>:<duration>}+`

Specifies the mode(s) and duration(s) for the specified scenario.

Supported Families

See the Tcl Commands and Supported Families table for a list of supported families.

Examples

This example creates a new scenario called myscenario:

```
smartpower_add_new_scenario -name "MyScenario" -mode "Custom_1:50.00"
"Custom_2:25.00" -mode "Active:25.00"
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_add_pin_in_domain

Tcl command; adds a pin into a clock or set domain.

```
smartpower_add_pin_in_domain -pin_name {pin_name} -pin_type {value} –domain_name
{domain_name} -domain_type {value}
```

Arguments

`-pin_name {pin_name}`

Specifies the name of the pin to add to the domain.

`-pin_type {value}`

Specifies the type of the pin to add. The following table shows the acceptable values for this argument:

Value	Description
clock	The pin to add is a clock pin
data	The pin to add is a data pin

`-domain_name {domain_name}`

Specifies the name of the domain in which to add the specified pin.

`-domain_type {value}`

Specifies the type of domain in which to add the specified pin. The following table shows the acceptable values for this argument:

Value	Description
clock	The domain is a clock domain
set	The domain is a set domain

Supported Families

See the Tcl Commands and Supported Families table for a list of supported families.

Notes

- The `domain_name` must be a name of an existing domain.
- The `pin_name` must be a name of a pin that exists in the design.

Examples

The following example adds a clock pin to an existing Clock domain:

```
smartpower_add_pin_in_domain -pin_name { XCMP3/U0/U1:Y } -pin_type {clock} –domain_name {clk1} -domain_type {clock}
```

The following example adds a data pin to an existing Set domain:

```
smartpower_add_pin_in_domain -pin_name {XCMP3/U0/U1:Y} -pin_type {data} -domain_name {myset} -domain_type {set}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

`smartpower_remove_pin_of_domain`

smartpower_battery_settings

This SmartPower Tcl command sets the battery capacity in SmartPower. The battery capacity is used to compute the battery life of your design.

```
smartpower_battery_settings -capacity {decimal value}
```

Parameters

`-capacity {decimal value}`

Value must be a positive decimal.

This parameter is mandatory.

Exceptions

None

Returns

This command does not return a value.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Usage

This section parameters for the command, their types, and the values they can be set to.

smartpower_battery_settings	Type	Value	Description
capacity	Decimal	Positive decimal	Specify the battery capacity in mA*Hours

Example

This example sets the battery capacity to 1800 mA * Hours.

```
smartpower_battery_settings –capacity {1800}
```

smartpower_change_clock_statistics

Tcl command; changes the default frequencies and probabilities for a specific domain.

```
smartpower_change_clock_statistics -domain_name {value} -clocks_freq {value} -
clocks_proba {value} -registers_freq {value} -registers_proba {value} -set_reset_freq
{value} -set_reset_proba {value} -primaryinputs_freq {value} -primaryinputs_proba {value} -
combinational_freq {value} -combinational_proba {value}
```

Arguments

`-domain_name{value}`

Specifies the domain name in which to initialize frequencies and probabilities.

`-clocks_freq {value}`

Specifies the user input frequency in Hz, KHz, or MHz for all clocks.

`-clocks_proba {value}`

Specifies the user input probability in % for all clocks.

`-registers_freq {value}`

Specifies the user input frequency (in Hz, KHz, or MHz) or the toggle rate (in %). If the unit is not provided and toggle rate is active, the value is handled as a toggle rate; if toggle rate is not active, the value is handled as a frequency.

`-registers_proba {value}`

Specifies the user input probability in % for all registers.

`-set_reset_freq {value}`

Specifies the user input frequency (in Hz, KHz, or MHz) or the toggle rate (in %). If the unit is not provided and toggle rate is active, the value is handled as a toggle rate; if toggle rate is not active, the value is handled as a frequency.

`-set_reset_proba {value}`

Specifies the user input probability in % for all set/reset nets.

`-primaryinputs_freq {value}`

Specifies the user input frequency (in Hz, KHz, or MHz) or the toggle rate (in %). If the unit is not provided and toggle rate is active, the value is handled as a toggle rate; if toggle rate is not active, the value is handled as a frequency.

`-primaryinputs_proba {value}`

Specifies the user input probability in % for all primary inputs.

`-combinational_freq {value}`

Specifies the user input frequency (in Hz, KHz, or MHz) or the toggle rate (in %). If the unit is not provided and toggle rate is active, the value is handled as a toggle rate; if toggle rate is not active, the value is handled as a frequency.

`-combinational_proba {value}`

Specifies the user input probability in % for all combinational combinational output.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all clocks withs:

```
smartpower_change_clock_statistics -domain_name {my_domain} -clocks_freq {10 MHz} -
clocks_proba {20} -registers_freq {10 MHz} -registers_proba {20} -set_reset_freq {10
MHz} -set_reset_proba {20} -primaryinputs_freq {10 MHz} -primaryinputs_proba {20} -
combinational_freq {10 MHz} -combinational_proba {20}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_change_setofpin_statistics

Tcl command; changes the default frequencies and probabilities for a specific set.

```
smartpower_change_setofpin_statistics -domain_name {value} -data_freq {value} -
data_proba {value}
```

Arguments

`-domain_name{value}`

Specifies the domain name in which to initialize data frequencies and probabilities.

`-data_freq {value}`

Specifies the user input data frequency in Hz, KHz, or MHz for all sets of pins.

`-data_proba {value}`

Specifies the user input data probability in % for all sets of pins.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all clocks withs:

```
smartpower_change_setofpin_statistics -domain_name {my_domain} -data_freq {10 MHz} -
data_proba {20}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_commit

Tcl command; saves the changes to the design (.adb) file.

```
smartpower_commit
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
smartpower_commit
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_restore

smartpower_compute_vectorless

This Tcl command executes a vectorless analysis of the current operating mode.

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
smartpower_compute_vectorless
```

See Also

Tcl Command Documentation Conventions

Designer Tcl Command Reference

smartpower_create_domain

Tcl command; creates a new clock or set domain.

```
smartpower_create_domain -domain_type {value} -domain_name {domain_name}
```

Arguments

`-domain_type {value}`

Specifies the type of domain to create. The following table shows the acceptable values for this argument:

Value	Description
clock	The domain is a clock domain
set	The domain is a set domain

`-domain_name {domain_name}`

Specifies the name of the new domain.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

The `domain name` cannot be the name of an existing domain.

The domain `type` must be either clock or set.

Examples

The following example creates a new clock domain named "clk2":

```
smartpower_create_domain -domain_type {clock} -domain_name {clk2}
```

The following example creates a new set domain named "myset":

```
smartpower_create_domain -domain_type {set} -domain_name {myset}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

`smartpower_remove_domain`

smartpower_edit_scenario

Tcl command; edits a scenario.

```
smartpower_edit_scenario -name {value} -description {value} -mode {value} -new_name {value}
```

Arguments

`-name {value}`

Specifies the name of the scenario.

`-description {value}`

Specifies the description of the scenario.

`-mode {<operating mode>:<duration>}`

Specifies the mode(s) and duration(s) for the specified scenario.

`-new_name {value}`

Specifies the new name for the scenario

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example edits the name of myscenario to finalscenario:

```
smartpower_edit_scenario -name myscenario -new_name finalscenario
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_import_vcd

This SmartPower Tcl command imports into SmartPower a VCD file generated by a simulation tool. SmartPower extracts the frequency and probability information from the VCD.

```
import_vcd -file "VCD file" [-opmode "mode name"] [-with_vectorless "TRUE | FALSE"] [-
partial_parse\ "TRUE | FALSE"] [-start_time "decimal value"] [-end_time "decimal value"]
\
[-auto_detect_top_level_name "TRUE | FALSE"] [-top_level_name "top level name"] [-
glitch_filtering\ "false | auto | true"] [-glitch_threshold "integer value"] [-stop_time
"decimal value"]
```

Parameters

`-file "VCD file"`

Value must be a file path. This parameter is mandatory.

`[-opmode "mode name"]`

Value must be a string. This parameter is optional.

`[-with_vectorless "TRUE | FALSE"]`

Value must be a boolean. This parameter is optional.

`[-partial_parse "TRUE | FALSE"]`

Value must be a boolean. This parameter is optional.

`[-start_time "decimal value"]`

Value must be a positive decimal. This parameter is optional.

`[-end_time "decimal value"]`

Value must be a positive decimal. This parameter is optional.

`[-auto_detect_top_level_name "TRUE | FALSE"]`

Value must be a boolean. This parameter is optional.

`[-top_level_name "top level name"]`

Value must be a string. This parameter is optional.

`[-glitch_filtering "false | auto | true"]`

Value must be one of false | auto | true. This parameter is optional.

`[-glitch_threshold "integer value"]`

Value must be a positive integer. This parameter is optional.

Exceptions

None

Returns

This command does not return a value.

Usage

This section lists all the parameters for the command, their types, and the values they can be set to. The default value is always listed first.

smartpower_import_vcd	Type	Values	Description
file	String	Path to a VCD file	Path to a VCD file.
opmode	String	Operating mode name “Active” by default	Operating mode in which the VCD will be imported. If the mode doesn’t exist, it will be created.
with_vectorless	Boolean	TRUE|FALSE	Specify the method to set the frequency and probability information for signals not annotated by the VCD TRUE: use the vectorless analysis FALSE: use average value computed from the VCD.
partial_parse	Boolean	FALSE|TRUE	Enable partial parsing of the VCD. Start time and end time need to be specified when TRUE.
start_time	Decimal value	positive decimal nanoseconds (ns)	Specify the starting timestamp of the VCD extraction in ns. It must be lower than the specified end_time. It must be lower than the last timestamp in the VCD file.
end_time	Decimal value	positive decimal nanoseconds (ns)	Specify the end timestamp of the VCD extraction in ns. It must be higher than the specified start_time.
auto_detect_top_level_name	Boolean	TRUE|FALSE	Enable the auto detection of the top level name in the VCD file. Top_level_name needs to be specified when FALSE .
top_level_name	Boolean	Full hierarchical name	Specify the full hierarchical name of the instance of the design in the VCD file.
glitch_filtering	Boolean	Auto|FALSE|TRUE	AUTO: Enable glitch filtering with predefined thereshold based on the family TRUE: Enable glitch filtering, glitch_threshold must be specified FALSE: Disable glitch

smartpower_import_vcd	Type	Values	Description
			filtering.
glitch_threshold	Integer	Positive integer	Specify the threshold in ps below which glitches are filtered out.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

The Tcl command below imports the power.vcd file generated by the simulator into SmartPower:

```
smartpower_import_vcd –file “../../simulation/power.vcd”
```

The Tcl command below extracts information between 1ms and 2ms in the simulation, and stores the information into a custom mode:

```
smartpower_import_vcd –file “../../simulation/power.vcd” –partial_parse TRUE –start_time
1000000 –end_time 2000000 –opmode “power_1ms_to_2ms”
```

smartpower_init_do

Tcl command; initializes the frequencies and probabilities for clocks, registers, set/reset nets, primary inputs, combinational outputs, enables and other sets of pins, and selects a mode for initialization.

```
smartpower_init_do -with {value} -opmode {value} -clocks {value} -registers {value} -
set_reset {value} -primaryinputs {value} -combinational {value} -enables {value} -othersets
{value}
```

Arguments

`-with{value}`

This sets the option of initializing frequencies and probabilities with vectorless analysis or with fixed values. The following table shows the acceptable values for this argument:

Value	Description
vectorless	Initializes frequencies and probabilities with vectorless analysis
fixed	Initializes frequencies and probabilities with fixed values

`-opmode {value}`

Optional; specifies the mode in which to initialize frequencies and probabilities. The value must be Active or Flash*Freeze.

`-clocks {value}`

This sets the option of initializing frequencies and probabilities for all clocks. The following table shows the acceptable values for this argument:

Value	Description
true	Initializes frequencies and probabilities for all clocks

Value	Description
false	Does not initialize frequencies and probabilities for all clocks

`-registers {value}`

This sets the option of initializing frequencies and probabilities for all registers. The following table shows the acceptable values for this argument:

Value	Description
true	Initializes frequencies and probabilities for all registers
false	Does not initialize frequencies and probabilities for all registers

`-set_reset {value}`

This sets the option of initializing frequencies and probabilities for all set/reset nets. The following table shows the acceptable values for this argument:

Value	Description
true	Initializes frequencies and probabilities for all set/reset nets
false	Does not initialize frequencies and probabilities for all set/reset nets

`-primaryinputs{value}`

This sets the option of initializing frequencies and probabilities for all primary inputs. The following table shows the acceptable values for this argument:

Value	Description
true	Initializes frequencies and probabilities for all primary inputs
false	Does not initialize frequencies and probabilities for all primary inputs

`-combinational {value}`

This sets the option of initializing frequencies and probabilities for all combinational outputs. The following table shows the acceptable values for this argument:

Value	Description
true	Initializes frequencies and probabilities for all combinational outputs
false	Does not initialize frequencies and probabilities for all combinational outputs

`-enables {value}`

This sets the option of initializing frequencies and probabilities for all enable sets of pins. The following table shows the acceptable values for this argument:

Value	Description
true	Initializes frequencies and probabilities for all enable sets of pins

Value	Description
false	Does not initialize frequencies and probabilities for all enable sets of pins

`-othersets {value}`

This sets the option of initializing frequencies and probabilities for all other sets of pins. The following table shows the acceptable values for this argument:

Value	Description
true	Initializes frequencies and probabilities for all other sets of pins
false	Does not initialize frequencies and probabilities for all other sets of pins

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all clocks with:

```
smartpower_init_do -with {vectorless} -opmode {my_mode} -clocks {true} -registers {true} -asynchronous {true} -primaryinputs {true} -combinational {true} -enables {true} -othersets {true}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_init_set_clocks_options

Tcl command; initializes the clock frequency options of all clock domains.

```
smartpower_init_set_clocks_options -with_clock_constraints {value} -with_default_values {value} -freq {value} -duty_cycle {value}
```

Arguments

`-with_clock_constraints {value}`

This sets the option of initializing the clock frequencies with frequency constraints from SmartTime. The following table shows the acceptable values for this argument:

Value	Description
true	Sets initialize clock frequencies with clock constraints ON
false	Sets initialize clock frequencies with clock constraints OFF

`-with_default_values {value}`

This sets the option of initializing the clock frequencies with a user input default value. The following table shows the acceptable values for this argument:

Value	Description
true	Sets initialize clock frequencies with default values ON
false	Sets initialize clock frequencies with default values OFF

`-freq {value}`

Specifies the user input frequency in Hz, KHz, or MHz.

`-duty_cycle {value}`

Specifies the user input duty cycles in %.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all clocks after executing `smartpower_init_do` with `-clocks {true}`:

```
smartpower_init_set_clocks_options -with_clock_constraints {true} -with_default_values
{true} -freq {10 MHz} -duty_cycle {20}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_init_set_combinational_options

Tcl commands; initializes the frequency and probability of all combinational outputs.

```
smartpower_init_set_combinational_options -freq {value} -proba {value}
```

Arguments

`-freq {value}`

Specifies the user input frequency (in Hz, KHz, or MHz) or the toggle rate (in %). If the unit is not provided and toggle rate is active, the value is handled as a toggle rate; if toggle rate is not active, the value is handled as a frequency.

`-proba {value}`

Specifies the user input probability in %.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all combinational signals after executing `smartpower_init_do` with `-combinational {true}`:

```
smartpower_init_set_combinational_options -freq {10 MHz} -proba {20}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_init_set_enables_options

Tcl command; initializes the clock frequency of all enable clocks with the initialization options.

```
smartpower_init_set_enables_options -freq {value} -proba {value}
```

Arguments

`-freq {value}`

Specifies the user input frequency (in Hz, KHz, or MHz).

`-proba {value}`

Specifies the user input probability in %.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all clocks after executing `smartpower_init_do` with `-enables {true}`:

```
smartpower_init_set_enables_options -freq {10 MHz} -proba {20}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_init_set_primaryinputs_options

Tcl command; initializes the frequency and probability of all primary inputs.

```
smartpower_init_set_primaryinputs_options -freq {value} -proba {value}
```

Arguments

`-freq {value}`

Specifies the user input frequency (in Hz, KHz, or MHz) or the toggle rate (in %). If the unit is not provided and toggle rate is active, the value is handled as a toggle rate; if toggle rate is not active, the value is handled as a frequency.

`-proba {value}`

Specifies the user input probability in %.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all primary inputs after executing `smartpower_init_do` with `-primaryinputs {true}`:

```
smartpower_init_set_primaryinputs_options -freq {10 MHz} -proba {20}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_init_set_registers_options

Tcl command; initializes the frequency and probability of all register outputs.

```
smartpower_init_set_registers_options -freq {value} -proba {value}
```

Arguments

`-freq {value}`

Specifies the user input frequency (in Hz, KHz, or MHz) or the toggle rate (in %). If the unit is not provided and toggle rate is active, the value is handled as a toggle rate; if toggle rate is not active, the value is handled as a frequency.

`-proba {value}`

Specifies the user input probability in %.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Exceptions

None

Examples

The following example initializes all register outputs after executing `smartpower_init_do` with `-registers {true}`:

```
smartpower_init_set_registers_options -freq {10 MHz} -proba {20}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_init_setofpins_values

Tcl command; initializes the frequency and probability of all sets of pins.

```
smartpower_init_setofpins_values -domain_name {name} -freq {value} -proba {value}
```

Arguments

`-domain_name{name}`

Specifies the set of pins that will be initialized. The following table shows the acceptable values for this argument:

Value	Description
IOsEnableSet	Specifies that the IOsEnableSet set of pins will be initialized
MemoriesEnableSet	Specifies that the MemoriesEnableSet set of pins will be initialized

`-freq {value}`

Specifies the user input frequency in Hz, MHz, or KHz.

`-proba {value}`

Specifies the user input probability in %.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all primary inputs after executing `smartpower_init_do` with `-othersets {true}`:

```
smartpower_init_setofpins_values -domain_name {IOsEnableSet} -freq {10 MHz} -proba {20}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_remove_all_annotations

Tcl command; removes all initialization annotations for the specified mode.

```
smartpower_remove_all_annotations -opmode {value}
```

Arguments

`-opmode {value}`

Removes all initialization annotations for the specified mode, where value must be Active or Flash*Freeze.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all clocks with opmode Acitve:

```
smartpower_remove_all_annotations -opmode {Active}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_remove_file

Tcl command; removes a VCD file from the specified mode or all operating mode. Frequency and probability information of signals annotated by the VCD are set back to the default value..

```
remove_file
-file {value} \
-format {value} \
-opmode {value} \
```

Arguments

`-file {value}`

Specifies the file to be removed. This is mandatory.

`-format VCD`

Specifies that the type to be removed is a VCD file. This is mandatory.

`[-opmode {value}]`

Specifies the operating mode. This is optional. The following table shows the acceptable values for this argument:

Value	Description
Active	The operating mode is set to active
Standby	The operating mode is set to static
Flash*Freeze	The operating mode is set to Flash*Freeze

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example removes the file test.vcd from the Active mode.

```
smartpower_remove_file -file "test.vcd" -format VCD -opmode "Active"
```

This example removes the VCD file power1.vcd from all operating modes:

```
smartpower_remove_file –file “power1.vcd” –format VCD
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

This command was obsoleted in SmartPower v8.5. Update your script to use `smartpower_remove_pin_probability` to remove the pin probability.

Note: Note: The information below is obsolete and should only be used as reference when executing previously-created scripts. Update your scripts to use `smartpower_remove_pin_probability`.

Removes the probability value associated with a specific pin. This pin will have a default probability based on the domain set it belongs to.

```
smartpower_remove_pin_enable_rate –pin_name {pin_name}
```

Arguments

`-pin_name {pin_name}`

Specifies the name of the pin with the probability to remove. This pin must be the direct driver of an enable pin.

Supported Families

SmartFusion, IGLOO, ProASIC3, Fusion

Exceptions

None

Examples

The following example removes the probability of the pin driving the enable pin of a bidirectional I/O:

```
Smartpower_remove_pin_enable_rate –pin_name mybibuf/U0/U1:EOUT
```

smartpower_remove_pin_probability

Tcl command; removes the probability value associated with a specific pin. This pin will have a default probability based on the domain set it belongs to.

```
smartpower_remove_pin_probability –pin_name {pin_name}
```

Arguments

`-pin_name {pin_name}`

Specifies the name of the pin with the probability to remove. This pin must be the direct driver of an enable pin.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

The following example removes the probability of the pin driving the enable pin of a bidirectional I/O:

```
Smartpower_remove_pin_probability –pin_name mybibuf/U0/U1:EOUT
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_set_pin_probability

smartpower_remove_scenario

Tcl command; removes a scenario from the current design.

```
smartpower_remove_scenario -name {value}
```

Arguments

`-name {value}`

Specifies the name of the scenario.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example removes a scenario from the current design:

```
smartpower_remove_scenario -name myscenario
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_set_mode_for_analysis

Tcl command; sets the mode for cycle-accurate power analysis.

```
smartpower_set_mode_for_analysis -mode {value}
```

Arguments

`-mode {value}`

Specifies the mode for cycle-accurate power analysis.

Value	Description
Active	The operating mode is set to Active
Standby	The operating mode is set to Standby
Flash*Freeze	The operating mode is set to Flash*Freeze

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

The following example sets the mode for analysis to active:

```
smartpower_set_mode_for_analysis -mode {active}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_set_mode_for_pdpr

This SmartPower Tcl command sets the operating mode used by the Power Driven Place and Route (PDPR) tool during power optimization.

```
smartpower_set_mode_for_pdpr -opmode { value}
```

Parameters

`-opmode {value}`

Value must be a valid operating mode.

This parameter is mandatory.

Sets the operating mode for your power driven place and route.

Exceptions

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Return Value

This command does not return a value.

Examples

This example sets the Active mode as the operating mode for Power Driven Place and Route.

```
set_mode_for_pdpr -opmode "Active"
```

This example creates a custom mode and set it to be used by Power Driven Place and Route (PDPR).

```
smartpower_add_new_custom_mode –name “MyCustomMode” \
–description “for PDPR” –base_mode “Active”
smartpower_set_mode_for_pdpr –opmode “MyCustomMode
```

See Also

Tcl Command Documentation Conventions

smartpower_set_operating_condition

Tcl command; sets the operating conditions used in SmartPower to one of the pre-defined types.

```
smartpower_set_operating_condition -opcond {value}
```

Arguments

`-opcond {value}`

Specifies the value of the operating condition. The following table shows the acceptable values for this argument:

Value	Description

Value	Description
best	Sets the operating conditions to best
typical	Sets the operating conditions to typical
worst	Sets the operating conditions to worst

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example sets the operating conditions to best:

```
smartpower_set_operating_condition -opcond {best}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

This command was obsoleted in SmartPower v8.5. Update your script to use `smartpower_set_pin_probability` to set the pin probability.

Note: The information below is obsolete and should only be used as reference when executing previously-created scripts. Update your scripts to use smartpower_set_pin_probability.

Enables you to set the probability value of a pin driving an enable pin. For I/Os, if you do not use this command, the probability of the IOEnableSet is used. For memories, if you do not use this command, the probability of the MemoriesEnableSet is used.

```
smartpower_set_pin_enable_rate -pin_name {pin_name} –pin_enable_rate {value}
```

Arguments

`-pin_name {pin_name}`

Specifies the name of a pin for which the probability will be set. This pin must be the direct driver of an enable pin.

`-pin_enable_rate {value}`

Specifies the value of the pin probability as a percentage, which can be any positive decimal between 0 and 100, inclusive.

Supported Families

SmartFusion, IGLOO, ProASIC3, Fusion

Exceptions

None

Examples

The following example sets the probability of the pin driving the enable pin of a bidirectional I/O

```
smartpower_set_pin_enable_rate -pin_name mybibuf/U0/U1:EOUT   \
–pin_enable_rate 50.4
```

smartpower_set_pin_probability

Enables you to set the probability value of a pin driving an enable pin. For I/Os, if you do not use this command, the probability of the IOEnableSet is used. For memories, if you do not use this command, the probability of the MemoriesEnableSet is used.

```
smartpower_set_pin_probability -pin_name {pin_name} –pin_enable_rate {value}
```

Arguments

`-pin_name {pin_name}`

Specifies the name of a pin for which the probability will be set. This pin must be the direct driver of an enable pin.

`-pin_proba {value}`

Specifies the value of the pin probability as a percentage, which can be any positive decimal between 0 and 100, inclusive.

Supported Families

SmartFusion, IGLOO, ProASIC3, Fusion

Notes

- None

Exceptions

- None

Examples

The following example sets the probability of the pin driving the enable pin of a bidirectional I/O

```
smartpower_set_pin_probability -pin_name mybibuf/U0/U1:EOUT   \
–pin_proba 50.4
```

See Also

smartpower_remove_pin_probability

smartpower_set_preference

Tcl command; sets the following preferences: power unit, frequency unit, operating mode, operating conditions, and toggle. These preferences can also be set from the preferences dialog box.

```
smartpower_set_preference -powerunit {value} -frequnit {value} -opmode {value} -opcond
{value} -toggle {value}
```

Arguments

`-powerunit {value}`

Specifies the unit in which power is set. The following table shows the acceptable values for this argument:

Value	Description
W	The power unit is set to watts
mW	The power unit is set to milliwatts

Value	Description
uW	The power unit is set to microwatts

`-frequnit {value}`

Specifies the unit in which frequency is set. The following table shows the acceptable values for this argument:

Value	Description
Hz	The frequency unit is set to hertz
kHz	The frequency unit is set to kilohertz
MHz	The frequency unit is set to megahertz

`-opmode {value}`

Specifies the operating mode. The following table shows the acceptable values for this argument:

Value	Description
active	The operating mode is set to active
static	The operating mode is set to static
sleep	The operating mode is set to sleep
Flash*Freeze	The operating mode is set to Flash*Freeze
shutdown	The operating mode is set to shutdown

`-opcond {value}`

Specifies the operating condition. The following table shows the acceptable values for this argument:

Value	Description
worst	The operating condition is set to worst case
typical	The operating condition is set to typical case
best	The operating condition is set to best case

`-toggle {value}`

Specifies the toggle. The following table shows the acceptable values for this argument:

Value	Description
true	The toggle is set to true
false	The toggle is set to false

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

- The following arguments have been removed. Running the script will trigger a warning message: Warning: Invalid argument: -argname "argvalue" Ignored. Ignore the warning.

-maxblocks {integer > 0}

-maxpins [{integer > 0}

-sortorder {ascending, descending}

-sortby {powervalues, alphabetical}

- Flash*Freeze, Sleep, and Shutdown are available only for certain families and devices.
- Worst and Best operating conditions are available only for certain families and devices.

Examples

This example sets the frequency of the power unit to "watts", the frequency unit to "Hz", the operating mode to "active", the operating condition to "typical", and the toggle to "true":

```
smartpower_set_setpreferences -powerunit {w} -frequnit {hz} -opmode {active} -opcond {typical} -toggle {true}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

SmartPower Preferences

smartpower_set_scenario_for_analysis

Tcl command; sets the scenario for cycle-accurate power analysis.

```
smartpower_set_scenario_for_analysis -scenario{value}
```

Arguments

`-scenario {value}`

Specifies the mode for cycle-accurate power analysis.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

The following example sets the scenario for analysis to my_scenario:

```
smartpower_set_scenario_for_analysis -scenario {my_scenario}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_set_temperature_opcond

Tcl command; sets the temperature in the operating conditions to one of the pre-defined types.

```
smartpower_set_temperature_opcond -use{value}
```

Arguments

`-use{value}`

Specifies the temperature in the operating conditions. The following table shows the acceptable values for this argument:

Value	Description
oprange	Sets the temperature in the operating conditions as specified in your Project Settings.
design	Sets the temperature in the operating conditions as specified in the SmartPower design-wide operating range. Applies to SmartPower only.
mode	Sets the temperature in the operating conditions as specified in the SmartPower mode-specific operating range. Applies to SmartPower only.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example sets the temperature in the operating conditions as specified in the custom mode-settings:

```
smartpower_set_temperature_opcond -use{mode}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_set_voltage_opcond

Tcl command; sets the voltage in the operating conditions.

```
smartpower_set_voltage_opcond -voltage{value} -use{value}
```

Arguments

`-voltage{value}`

Specifies the voltage supply in the operating conditions. The following table shows the acceptable values for this argument:

Value	Description
VDD	Sets the voltage operating conditions for VDD
VDDI 2.5	Sets the voltage operating conditions fo r VDDI 2.5
VPP	Sets the voltage operating conditions for VPP

`-use{value}`

Specifies the voltage in the operating conditions for each voltage supply. The following table shows the acceptable values for this argument:

Value	Description
oprange	Sets the voltage in the operating conditions as specified in your Project Settings.
design	Sets the voltage in the operating conditions as specified in the SmartPower design-wide operating range. Applies to SmartPower only.
mode	Sets the voltage in the operating conditions as specified in the SmartPower mode-specific operating range. Applies to SmartPower only.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example sets the VCCA as specified in the SmartPower mode-specific settings:

```
smartpower_set_voltage_opcond -voltage{vcca} -use{mode}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_temperature_opcond_set_design_wide

Tcl command; sets the temperature for SmartPower design-wide operating conditions.

```
smartpower_temperature_opcond_set_design_wide -best{value} -typical{value} -worst{value} -
thermal_mode{value}
```

Arguments

`-best{value}`

Specifies the best temperature (in degrees Celsius) used for design-wide operating conditions.

`-typical{value}`

Specifies the typical temperature (in degrees Celsius) used for design-wide operating conditions.

`-worst{value}`

Specifies the worst temperature (in degrees Celsius) used for design-wide operating conditions.

`-thermal_mode{value}`

Specifies the mode in which the junction temperature is computed. The following table shows the acceptable values for this argument:

Value	Description
ambient	The junction temperature will be iteratively computed with total static power
opcond	The junction temperature will be given as one of the operating condition range values specified in the device selection

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example sets the temperature for design-wide operating conditions to Best 20, Typical 30, and Worst 60:

```
smartpower_temperature_opcond_set_design_wide -best{20} -typical{30} -worst{60}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_temperature_opcond_set_mode_specific

Tcl command; sets the temperature for SmartPower mode-specific operating conditions.

```
smartpower_temperature_opcond_set_mode_specific -opmode{value} -thermal_mode{value} -
best{value} -typical{value} -worst{value} -thermal_mode{value}
```

Arguments

`-opmode {value}`

Specifies the operating mode. The following table shows the acceptable values for this argument:

Value	Description
Active	The operating mode is set to Active
Standby	The operating mode is set to Standby
Flash*Freeze	The operating mode is set to Flash*Freeze

`-thermal_mode{value}`

Specifies the mode in which the junction temperature is computed. The following table shows the acceptable values for this argument:

Value	Description
ambient	The junction temperature will be iteratively computed with total static power
opcond	The junction temperature will be given as one of the operating condition range values specified in the device selection

`-best{value}`

Specifies the best temperature (in degrees Celsius) for the selected mode.

`-typical{value}`

Specifies the typical temperature (in degrees Celsius) for the selected mode.

`-worst{value}`

Specifies the worst temperature (in degrees Celsius) for the selected mode.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example sets the temperature for mode-specific operating conditions for mode1:

```
smartpower_temperature_opcond_set_mode_specific -mode{mode1} -best{20} -typical{30} -worst{60}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_voltage_opcond_set_design_wide

Tcl command; sets the voltage settings for SmartPower design-wide operating conditions.

```
smartpower_voltage_opcond_set_design_wide -voltage{value} -best{value} -typical{value} -worst{value}
```

Arguments

`-voltage{value}`

Specifies the voltage supply in the operating conditions. The following table shows the acceptable values for this argument:

Value	Description
VDD	Sets the voltage operating conditions for VDD
VDDI 2.5	Sets the voltage operating conditions for VDDI 2.5
VPP	Sets the voltage operating conditions for VPP
VCCA	Sets the voltage operating conditions for VCCA
VCCI 3.3	Sets the voltage operating conditions for VCCI 3.3
VCCI 2.5	Sets the voltage operating conditions for VCCI 2.5
VCCI 1.8	Sets the voltage operating conditions for VCCI 1.8
VCCI 1.5	Sets the voltage operating conditions for VCCI 1.5
VCC33A	Sets the voltage operating conditions for VCC33A
VCCDA	Sets the voltage operating conditions for VCCDA

`-best{value}`

Specifies the best voltage used for design-wide operating conditions.

`-typical{value}`

Specifies the typical voltage used for design-wide operating conditions.

`-worst{value}`

Specifies the worst voltage used for design-wide operating conditions.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example sets VCCA for design-wide to best 20, typical 30 and worst 40:

```
smartpower_voltage_opcond_set_design_wide -voltage{VCCA} -best{20} -typical{30} -
worst{40}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_voltage_opcond_set_mode_specific

Tcl command; sets the voltage settings for SmartPower mode-specific use operating conditions.

```
smartpower_voltage_opcond_set_mode_specific -opmode{value} -voltage{value} -best{value} -
typical{value} -worst{value}
```

Arguments

`-opmode {value}`

Use this option to specify the mode from which the operating conditions are extracted to generate the report.

Value	Description
Active	The operating mode is set to Active
Standby	The operating mode is set to Standby
Flash*Freeze	The operating mode is set to Flash*Freeze

`-voltage{value}`

Specifies the voltage in the operating conditions. The following table shows the acceptable values for this argument:

Value	Description
VDD	Sets the voltage operating conditions for VDD
VDDI 2.5	Sets the voltage operating conditions for VDDI 2.5
VPP	Sets the voltage operating conditions for VPP
VCCA	Sets the voltage operating conditions for VCCA
VCCI 3.3	Sets the voltage operating conditions for VCCI 3.3
VCCI 2.5	Sets the voltage operating conditions for VCCI 2.5
VCCI 1.8	Sets the voltage operating conditions for VCCI 1.8
VCCI 1.5	Sets the voltage operating conditions for VCCI 1.5

Value	Description
VCC33A	Sets the voltage operating conditions for VCC33A
VCCDA	Sets the voltage operating conditions for VCCDA

`-best{value}`

Specifies the best voltage used for mode-specific operating conditions.

`-typical{value}`

Specifies the typical voltage used for mode-specific operating conditions.

`-worst{value}`

Specifies the worst voltage used for mode-specific operating conditions.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example sets the voltage for the static mode and sets best to 20, typical to 30 and worst to 40:

```
smartpower_voltage_opcond_set_mode_specific -opmode{active} -voltage{VCCA} -best{20} -
typical{30} -worst{40}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

SmartFusion, IGLOO, ProASIC3, and Fusion – Designer Commands

add_probe (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; adds a probe to an internal net in your design, using the original name from the optimized netlist in your design. Also, this command must be used in conjunction with the generate_probes command to generate a probed ADB file (see example below).

You must complete layout before you use this command.

```
add_probe –net <net_name> [-pin <pin_name>] [-port <port_name>] [-assign_to_used_pin <TRUE|FALSE>]
```

Arguments

`-net <net_name>`

Name of the net you want to probe. You cannot probe HARDWIRED, POWER, or INTRINSIC nets.

`-pin <pin_name>`

Name of the package pin at which you want to put the net to be probed. Argument is optional; if unspecified the net is routed to any free package pin.

`-port <port_name>`

Name of the port you are adding. Argument is optional; if unspecified the default value is PROBE_<n>.

`-assign_to_used_pin <TRUE|FALSE>`

Probes a net on an already used pin. The net on the existing pin will be disconnected. Argument is optional; if unspecified the net can be only routed on unused pin.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The example below adds a probe to the net Count8_0/INV_0_Y on pin 7 and uses the port name PROBE_1, then generates the probe ADB file named test1.adb.

Note that generate_probes is a separate Tcl command.

```
add_probe -net Count8_0/INV_0_Y -assign_to_used_pin {FALSE} -pin {7} -port {PROBE_1}
generate_probes –save test1.adb
```

See Also

delete_probe

generate_probes

Generating a Probed Design

Generate Probed Design - Add Probe(s) Dialog Box

Designer Tcl Command Reference

all_inputs

Tcl command; returns an object representing all input and inout pins in the current design.

```
all_inputs
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Exceptions

You can only use this command as part of a –from, -to, or –through argument in the following Tcl commands: set_min_delay, set_max_delay, set_multicycle_path, and set_false_path.

Examples

```
set_max_delay -from [all_inputs] -to [get_clocks ck1]
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

all_outputs

Tcl command; returns an object representing all output and inout pins in the current design.

```
all_outputs
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Exceptions

You can only use this command as part of a –from, -to, or –through argument in the following Tcl commands: set_min_delay, set_max_delay, set_multicycle_path, and set_false_path.

Examples

```
set_max_delay -from [all_inputs] -to [all_outputs]
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

all_registers

Tcl command; returns an object representing register pins or cells in the current scenario based on the given parameters.

```
all_registers [-clock clock_name]
[-async_pins][-output_pins][-data_pins][-clock_pins]
```

Arguments

-clock *clock_name*

Specifies the name of the clock domain to which the registers belong. If no clock is specified, all registers in the design will be targeted.

-async_pins

Lists all register pins that are async pins for the specified clock (or all registers asynchronous pins in the design).

-output_pins

Lists all register pins that are output pins for the specified clock (or all registers output pins in the design).

-data_pins

Lists all register pins that are data pins for the specified clock (or all registers data pins in the design).

-clock_pins

Lists all register pins that are data pins for the specified clock (or all registers clock pins in the design).

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Exceptions

You can only use this command as part of a –from, -to, or –through argument in the following Tcl commands: set_min_delay, set_max_delay, set_multicycle_path, and set_false_path.

Examples

```
set_max_delay 2.000 -from { ff_m:CLK ff_s2:CLK } -to [all_registers -clock_pins -clock { ff_m:Q }]
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

are_all_source_files_current (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; audits all source files and determines whether or not they are out of date / imported into the workspace. Returns '1' if all source files are current Returns '0' if all source files are not current This command ignores the Audit settings in your ADB file.

```
are_all_source_files_current
```

Arguments

None

Supported Family

See the Tcl Commands and Supported Families table for the list of families that support this command.

Exceptions

The command will return an error if arguments are passed.

Example

The following code will determine if your source files are current.

```
are_all_source_files_current
```

See Also

get_out_of_date_files
is_source_file_current
Designer Tcl Command Reference

backannotate (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; equivalent to executing the Back-Annotate command from the Tools menu. You can export an SDF file, after layout, along with the corresponding netlist in the VHDL or Verilog format. These files are useful in backannotated timing simulation.

Best practice is to export both SDF and the corresponding VHDL/Verilog files. This will avoid name conflicts in the simulation tool.

Designer must have completed layout before this command can be invoked, otherwise the command will fail.

```
backannotate -name file_name -format format_type -language language-dir directory_name [-
netlist] [-pin] [-use_emd]
```

Arguments

`-name file_name`

Use a valid file name with this option. You can attach the file extension .sdf to the File_Name, otherwise the tool will append .sdf for you.

`-format format_type`

Only SDF format is available for back annotation

`-language language`

The supported Language options are:

Value	Description
VHDL93	For VHDL-93 style naming in SDF
VERILOG	For Verilog style naming in SDF

`-dir directory_name`

Specify the directory in which all the files will be extracted.

`-netlist`

Forces a netlist to be written. The netlist will be either in Verilog or VHDL.

`-pin`

Designer exports the pin file with this option. The .pin file extension is appended to the design name to create the pin file.

`-use_emd`

Enables Export Enhanced Min Delays for Best Case option in your backannotated file.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

Example 1 uses default arguments and exports SDF file for back annotation:

```
backannotate
```

Example 2 uses some of the options for VHDL:

```
backannotate -dir \
 {..\my_design_dir} -name "fanouttest_ba.sdf" -format "SDF" –language \ "VHDL93" –netlist
```

Example 3 uses some of the options for Verilog:

```
backannotate -dir \
 {..\design} -name "fanouttest_ba.sdf" -format "SDF" -language "VERILOG" \
-netlist
```

Example 4 enables you to catch exceptions and respond based on the success of backannotate operation:

```
If  { [catch { backannotate -name "fanouttest_ba" -format "SDF" } ]} {
              Puts “Back annotation failed”
              # Handle Failure
} else {
              Puts “Back annotation successful”
              # Proceed with other operations
}
```

Example 5 enables Export Enhanced Min Delays for Best Case:

```
backannotate -dir \ {..\my_design_dir} -name "fanouttest_ba.sdf" -format "SDF"
–language \ "VHDL93" –netlist -use_emd
```

See Also

Tcl command documentation conventions

Designer Tcl Command Reference

check_constraints

Tcl command; checks all timing constraints in the current scenario for validity. This command performs the same checks as when the constraint is entered through SDC or Tcl.

```
check_constraints
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
check_constraints
```

check_timing_constraints

Tcl command; checks all timing constraints in the current timing scenario for validity.

```
check_timing_constraints
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
check_timing_constraints
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

clone_scenario

Tcl command; creates a new timing scenario by duplicating an existing one. You must provide a unique name (that is, it cannot already be used by another timing scenario).

```
clone_scenario name -source origin
```

Arguments

name

Specifies the name of the new timing scenario to create.

`-source` *origin*

Specifies the source of the timing scenario to clone (copy). The source must be a valid, existing timing scenario.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

This command creates a timing scenario with the specified name, which includes a copy of all constraints in the original scenario (specified with the -source parameter). The new scenario is then added to the list of scenarios.

Example

```
clone_scenario scenario_A –source {Primary}
```

See Also

`create_scenario`

`delete_scenario`

Tcl documentation conventions

Designer Tcl Command Reference

close_design (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; closes the current design and brings Designer to a fresh state to work on a new design. This is equivalent to selecting the Close command from the File menu.

```
close_design
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
if  { [catch { close_design }] {
            Puts “Failed to close design”
            # Handle Failure
} else {
            puts “Design closed successfully”
            # Proceed with processing a new design
}
```

See Also

open_design
close_design
new_design
Designer Tcl Command Reference

compile (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; compile Tcl arguments available for SmartFusion, IGLOO, ProASIC3 and Fusion families.

```
compile
-pdc_abort_on_error value
-pdc_eco_display_unmatched_objects value
-pdc_eco_max_warnings value
-demote_globals value
-demote_globals_max_fanout value
-promote_globals value
-promote_globals_min_fanout value
-promote_globals_max_limit value
-localclock_max_shared_instances value
-localclock_buffer_tree_max_fanout value
-combine_register value
-delete_buffer_tree value
-delete_buffer_tree_max_fanout value
-report_high_fanout_nets_limit value
```

Block creation mode only:

```
-block_remove_ios value
-block_add_interface value
-block_add_interface_fanout value
```

Block instantiation mode only:

```
-block_placement_conflicts value
-block_routing_conflicts value
```

Arguments

`-pdc_abort_on_error value`

Changes the “Abort on PDC error” behavior. The following table shows the values for this argument:

Value	Description
ON	Stops the flow if any error is reported in reading your PDC file
OFF	Skips errors in reading your PDC file and just report them as warnings.

Default: ON

Note: The flow always stops in the following two cases (even if this option is OFF):

- If there is a Tcl error (for example, the command does not exist or the syntax of the command is incorrect)
- The assign_local_clock command for assigning nets to LocalClocks fails. This may happen if any floor planning DRC check fails, such as, region resource check, fix macro check (one of the load on the net is outside the LocalClock region). If such an error occurs, then the Compile command fails. Correct your PDC file to proceed.

`-pdc_eco_display_unmatched_objects` *`value`*

Displays netlist objects in PDC that are not found in the imported netlist during Compile ECO mode.The following table shows the values for this argument:

Value	Description
ON	Reports netlist objects not found in the current netlist when reading the internal ECO PDC constraints
OFF	Specifies not to report netlist objects not found in the current netlist when reading the internal ECO PDC constraints

Default: OFF

`-pdc_eco_max_warnings` *`value`*

Defines the maximum number of errors/warnings in Compile ECO mode.

The value is the maximum number of error/warning messages to be displayed in the case of reading ECO constraints.

Default: 10000

`-demote_globals` *`value`*

Enables/disables global clock demotion of global nets to regular nets. The following table shows the values for this argument:

Value	Description
OFF	Disables global demotion of global nets to regular nets
ON	Enables global demotion of global nets to regular nets

Default: OFF

`-demote_globals_max_fanout` *`value`*

Defines the maximum fanout value of a demoted net;where value is the maximum value
Default: 12

Note: A global net is not automatically demoted (assuming the option is on) if the resulting fanout of the demoted net (if it was demoted) is greater than the max fanout value. Best practice is to set the automatic global demotion so that it only acts on small fanout nets. Drive high fanout nets with a clock network in the design to improve routability and timing.

`-promote_globals` *`value`*

Enables/disables global clock promotion. The following table shows the values for this argument:

Value	Description
ON	Enables global promotion of nets to global clock network
OFF	Disables global promotion of nets to global clock network

Default: OFF

`-promote_globals_min_fanout value`

Defines the minimum fanout of a promoted net; where *value* is the minimum fanout of a promoted net. Default:200

`-promote_globals_max_limit value`

Defines the maximum number of nets to be automatically promoted to global The default value is 0. This is not the total number as nets need to satisfy the minimum fanout constraint to be promoted. The promote_globals_max_limit value does not include globals that may have come from either the netlist or PDC file (quadrant clock assignment or global promotion).

Note: Demotion of globals through PDC or Compile is done before automatic global promotion is done.

Note: You may exceed the number of globals present in the device if you have nets already assigned to globals or quadrants from the netlist or by using a PDC file. The automatic global promotion adds globals on what already exists in the design.

`-localclock_max_shared_instances value`

Defines the maximum number of shared instances allowed to perform the legalization. This option is also available for quadrant clocks.

value is the maximum number of instances allowed to be shared by 2 LocalClock nets assigned to disjoint regions to perform the legalization (default is 12, range is 0-1000). If the number of shared instances is set to 0, no legalization is performed.

Note: If you assign quadrant clocks to nets using MultiView Navigator, no legalization is performed.

`-localclock_buffer_tree_max_fanout value`

Defines the maximum fanout value used during buffer insertion for clock legalization. This option is also available for quadrant clocks.

Set *value* to 0 to disable this option and prevent legalization (default value is 12, range is 0-1000). If the value is set to 0, no buffer insertion is performed. If the value is set to 1, there will be one buffer inserted per pin.

`-combine_register value`

Combines registers at the I/O into I/O-Registers. The following table shows the values for this argument:

Value	Description
ON	Combines registers at the I/O into I/O-Registers
OFF	Does not optimize and combine registers at the I/O.

Default: OFF

`-delete_buffer_tree value`

Enables/disables buffer tree deletion on the global signals. The buffer and inverter are deleted. The following table shows the values for this argument:

Value	Description
ON	Enables buffer tree deletion from the netlist
OFF	Disables buffer tree deletion from the netlist

Default: OFF

`-delete_buffer_tree_max_fanout value`

Defines the maximum fanout of a net after buffer tree deletion;

value is the maximum value; the default value is 12.

Note: A net does not automatically remove its buffer tree (assuming the option is on) if the resulting fanout of the net (if the buffer tree was removed) is greater than the max fanout value. Best

practice is to set the automatic buffer tree deletion only so that acts on small fanout nets. Drive high fanout nets with a clock network in the design to improve routability and timing.

`-report_high_fanout_nets_limit value`

Enables flip-flop net sections in the compile report and defines the number of nets to be displayed in the high fanout.

Default: 10

Block creation mode only:

`-block_remove_ios value`

Removes I/Os, if any in the design. Possible values are shown in the table below:

Value	Description
ON	Removes I/Os from the block (if possible)
OFF	Leaves I/Os (if any) unchanged

`-block_add_interface value`

Adds buffers on ports in the block, no fanout limit. Values shown in the table below:

Value	Description
ON	Adds buffers on ports
OFF	Does not add any buffers to ports

`-block_add_interface_fanout value`

Adds buffers on ports in the block whose fanout is greater than <value>. This option is used in conjunction with the -block_add_interface option above.

Block instantiation mode only:

`-block_placement_conflicts value`

If there multiple blocks instantiated in your design, Designer uses the placement options to resolve the conflicts. Values shown in the table below:

Value	Description
ERROR	Compile errors out if any instance from a designer block is unplaced. This is the default option.
RESOLVE	If some instances get unplaced for any reason, the remaining non-conflicting elements are unplaced. In other words, if there are any conflicts, nothing from the block is kept.
KEEP	If some instances get unplaced for any reason, the non-conflicting elements remaining are preserved but not locked (you can move them).
LOCK	If some instances get unplaced for any reason, the remaining non-conflicting elements are preserved and locked.

`-block_routing_conflicts value`

If there multiple blocks instantiated in your design, Designer uses the routing options to resolve the conflicts. Values shown in the table below:

Value	Description
ERROR	Compile errors out if any preserved net routing in a designer block is deleted.
RESOLVE	If a nets' routing is removed for any reason, the routing for non-conflicting nets is also removed. In other words, if there are any conflicts, no routing from the block is kept
KEEP	If a nets routing is removed for any reason, the routing for the non-conflicting nets is preserved but not locked (so that they can be rerouted).
LOCK	If the routing is removed for any reason, the remaining non-conflicting nets are preserved and locked; they cannot be rerouted. This is the default option.

Exceptions

You cannot instantiate an ARM design and create a User Block.

Examples

```
compile \
    -pdc_abort_on_error "ON" \
    -pdc_eco_display_unmatched_objects "OFF" \
    -pdc_eco_max_warnings 10000 \
    -demote_globals "OFF" \
    -demote_globals_max_fanout 12 \
    -promote_globals "OFF" \
    -promote_globals_min_fanout 200 \
    -promote_globals_max_limit 0 \
    -localclock_max_shared_instances 12 \
    -localclock_buffer_tree_max_fanout 12 \
    -combine_register "OFF" \
    -delete_buffer_tree "OFF" \
    -delete_buffer_tree_max_fanout 12 \
    -report_high_fanout_nets_limit 10
```

See Also

Setting Compile Options

Designer Tcl Command Reference

create_clock

Tcl command; creates a clock constraint on the specified ports/pins, or a virtual clock if no source other than a name is specified.

```
create_clock -period period_value [-name clock_name]
[-waveform> edge_list][source_objects]
```

Arguments

`-period` *`period_value`*

Specifies the clock period in nanoseconds. The value you specify is the minimum time over which the clock waveform repeats. The period_value must be greater than zero.

`-name` *`clock_name`*

Specifies the name of the clock constraint. You must specify either a clock name or a source.

`-waveform` *`edge_list`*

Specifies the rise and fall times of the clock waveform in ns over a complete clock period. There must be exactly two transitions in the list, a rising transition followed by a falling transition. You can define a clock starting with a falling edge by providing an edge list where fall time is less than rise time. If you do not specify -waveform option, the tool creates a default waveform, with a rising edge at instant 0.0 ns and a falling edge at instant (period_value/2)ns.

`source_objects`

Specifies the source of the clock constraint. The source can be ports, pins, or nets in the design. If you specify a clock constraint on a pin that already has a clock, the new clock replaces the existing one. You must specify either a source or a clock name.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Creates a clock in the current design at the declared source and defines its period and waveform. The static timing analysis tool uses this information to propagate the waveform across the clock network to the clock pins of all sequential elements driven by this clock source.

The clock information is also used to compute the slacks in the specified clock domain that drive optimization tools such as place-and-route.

Examples

The following example creates two clocks on ports CK1 and CK2 with a period of 6, a rising edge at 0, and a falling edge at 3:

```
create_clock -name {my_user_clock} -period 6 CK1
create_clock -name {my_other_user_clock} –period 6 –waveform {0 3} {CK2}
```

The following example creates a clock on port CK3 with a period of 7, a rising edge at 2, and a falling edge at 4:

```
create_clock –period 7 –waveform {2 4} [get_ports {CK3}]
```

See Also

create_generated_clock

Tcl Command Documentation Conventions

Designer Tcl Command Reference

create_generated_clock

Tcl command; creates an internally generated clock constraint on the ports/pins and defines its characteristics.

```
create_generated_clock [-name name] -source reference_pin [-divide_by divide_factor] [-
multiply_by multiply_factor] [-invert] source
```

Arguments

`-name` *`name`*

Specifies the name of the clock constraint.

`-source` *`reference_pin`*

Specifies the reference pin in the design from which the clock waveform is to be derived.

`-divide_by` *`divide_factor`*

Specifies the frequency division factor. For instance if the *divide_factor* is equal to 2, the generated clock period is twice the reference clock period.

`-multiply_by` *`multiply_factor`*

Specifies the frequency multiplication factor. For instance if the *multiply_factor* is equal to 2, the generated clock period is half the reference clock period.

`-invert`

Specifies that the generated clock waveform is inverted with respect to the reference clock.

`source`

Specifies the source of the clock constraint on internal pins of the design. If you specify a clock constraint on a pin that already has a clock, the new clock replaces the existing clock. Only one source is accepted. Wildcards are accepted as long as the resolution shows one pin.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Creates a generated clock in the current design at a declared source by defining its frequency with respect to the frequency at the reference pin. The static timing analysis tool uses this information to compute and propagate its waveform across the clock network to the clock pins of all sequential elements driven by this source.

The generated clock information is also used to compute the slacks in the specified clock domain that drive optimization tools such as place-and-route.

Examples

The following example creates a generated clock on pin U1/reg1:Q with a period twice as long as the period at the reference port CLK.

```
create_generated_clock -name {my_user_clock} –divide_by 2 –source [get_ports {CLK}]  U1/reg1:Q
```

The following example creates a generated clock at the primary output of myPLL with a period ¾ of the period at the reference pin clk.

```
create_generated_clock –divide_by 3 –multiply_by 4  -source clk [get_pins {myPLL:CLK1}]
```

See Also

`create_clock`

Tcl Command Documentation Conventions

Designer Tcl Command Reference

create_scenario

Tcl command; creates a new timing scenario with the specified name. You must provide a unique name (that is, it cannot already be used by another timing scenario).

```
create_scenario name
```

Arguments

`name`

Specifies the name of the new timing scenario.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

A timing scenario is a set of timing constraints used with a design. Scenarios enable you to easily refine the set of timing constraints used for Timing-Driven Place-and-Route, so as to achieve timing closure more rapidly.

This command creates an empty timing scenario with the specified name and adds it to the list of scenarios.

Example

```
create_scenario scenario_A
```

See Also

clone_scenario

Tcl Command Documentation Conventions

Designer Tcl Command Reference

delete_probe (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; deletes a probe on nets in a probed ADB file.

```
delete_probe –net <net_name>
```

Arguments

-net <*net_name*>

Name of the net you want to delete.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The example below deletes the probe on the net Count8_0/INV_0_Y.

```
delete_probe -net Count8_0/INV_0_Y
```

See Also

add_probe

Generating a Probed Design

Generate Probed Design - Add Probe(s) Dialog Box

Designer Tcl Command Reference

delete_scenario

Tcl command; deletes the specified timing scenario.

```
delete_scenario name
```

Arguments

name

Specifies the name of the timing scenario to delete.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

This command deletes the specified timing scenario and all the constraints it contains.

Exceptions

- At least one timing scenario must always be available. If the current scenario is the only one that exists, you cannot delete it.
- Scenarios that are linked to the timing analysis or layout cannot be deleted.

Example

```
delete_scenario scenario_A
```

See Also

create_scenario

Tcl Command Documentation Conventions

Designer Tcl Command Reference

export

Tcl command; saves your design to a file in the specified file format. The required and optional arguments this command takes depends on which file format you specify.

```
export
[-format value]
[-feature value]
[-secured_device value]
[-signature value]
[-pass_key value]
[-aes_key value]
[-from_config_file value]
[-number_of_devices value]
[-from_progfile_type value]
[-target_programmer value]
[-custom_security value]
[-fpga_security_level value]
[-from_security_level value]
[-security_permanent value]{filename}
[-from_program_pages value]
[-from_content value]
[-set_io_state value]
[-efm_block_security {location:X;security_level: value}]
[-efm_content {location:X;source:value}]
[-efm_block {location:X;config_file:{value}}]
[-efm_client {location:X;client: value;mem_file: value}}]
```

Arguments

`-format value`

Specifies the file format of the file to export. The exported files vary from one device family to another; see the Export help topic for a description of each file type and the list of supported families.

You can export the files listed in the table below using the value.

File Types	Value
Netlist Files	adl
	afl
	edn
	v
	vhd
Constraint Files	crt
	dcf
	gcf
	sdc
	pdc
	pin
Programming Files	afm
	bit
	bts_stp
	dc (exports a *.dat programming file)
	fus
	isc
	pdb
	1532
	svf
FlashPro Data File	fdb
Debugging Files	bsd
	prb
Timing Files	mod
	sdf
	stf
	tcl
Script Files	tcl

File Types	Value
Log Files	log
IBIS Files	ibs
Other Files	cob
	loc
	seg
Block Files	cdb
	cxf
	v
	vhd

`-feature {value}`

Select the silicon feature(s) you want to program. Possible values for this option are listed in the table below, or the instance-specific program options available only for specific families (as shown in the table below). Best practice is to specify your program parameters for each Embedded Flash Memory Block (EFMB) instance, from 0-3. The instance specific program options replace [-feature {*value*}].

value	Family
{setup_security:on/off}	SmartFusion
{prog_fpga:on/off}	SmartFusion
{prog_from:on/off}	SmartFusion
{prog_nvm:on/off}	SmartFusion
{setup_security}	Fusion
{prog_from}	Fusion
{all}	IGLOO; ProASIC3

In Tcl mode for Fusion, programming all features are turned off by default. If there is -feature {setup_security} or -feature {prog_from} the programming for the corresponding feature is activated.

In Tcl mode for SmartFusion, the programming option is read from the loaded PDB and then updated from the command if the is parameter specified. If programming of specific features is disabled, other parameters related to the feature programming are ignored. For example, if -feature {prog_fpga:off}, then -fdb_file and -fdb_source are ignored.

`-secured_device value`

Specifies whether the device you are programming is secured. You can specify yes or no to enable or disable secured programming.

`-signature value`

Optional argument that identifies and tracks Microsemi SoC designs and devices.

`-pass_key value`

Protects all the security settings for FPGA Array, FlashROM, and Embedded Flash Memory Block. The maximum length of this value is 32 characters. You must use hexadecimal characters for the pass key value.

`-aes_key` *`value`*

Decrypts FPGA Array and/or FlashROM and Embedded Flash Memory Block programming file content.

Max length is 32 HEX characters.

`-from_config_file` *`value`*

Specifies the location of the FlashROM configuration file.

`-number_of_devices` *`value`*

Specifies the number of devices you want to program. Applicable only when FlashROM has serialization regions.

`-from_progfile_type` *`value`*

Applicable only when FlashROM has serialization regions and STAPL file generation. Possible values:

Value	Description
single	Generates one programming file with all the generated incremental value(s) in the external source file
multiple	Generates one individual programming file for each generated incremental value(s) in the external source file

`-target_programmer` *`value`*

Applicable only when FlashROM has serialization regions and STAPL file generation. Possible values:

Value	Description
specific	Silicon Sculptor, BP Auto Programmer, or FlashPro
generic	Generic STAPL player

`-custom_security` *`value`*

Possible values:

Value	Description
yes	Custom security level
no	Standard security level

`-fpga_security_level` *`value`*

Possible values:

Value	Description
write_verify_protect	The security level is medium (standard) and the FPGA Array cannot be written or verified without a Pass Key
write_protect	The security level is write protected. The FPGA Array cannot be written without a Pass Key, but it is open for verification (custom FPGA)

Value	Description
encrypted	The security level is high (standard) and uses a 128-bit AES encryption
none	The FPGA Array can be written and verified without a Pass Key

`-from_security_level` *value*

Possible values:

Value	Description
write_verify_protect	The security level is medium (standard) and the FlashROM cannot be read, written or verified without a Pass Key
write_protect	The security level is write protected. The FlashROM cannot be written without a Pass Key, but it is open for reading and verification (custom FlashROM)
encrypted	The security level is high (standard) and uses a 128-bit AES encryption
none	The FlashROM can be written and verified without a Pass Key

`-security_permanent` *value*

Specifies whether the security settings for this file are permanent or not. Possible values:

Value	Description
yes	Permanently disable future modification of security settings for FPGA Array and FlashROM
no	Enable future modifications for FPGA Array and FlashROM

`-from_program_pages` *"value"*

Specifies FROM program pages in FlashPoint. If you use FlashROM content from an ADB file and do not specify a value, FlashPoint uses the same pages that were selected for programming in the previous FlashPoint session. Value may be a sequence of page numbers ("123") without a delimiter, or you can use any character or space as a delimiter, as in -from_program_pages "1 2 3".

You must specify pages for programming if you want FlashROM content from the UFC file.

`-from_content "`*value*`"`

Identifies the source file for the FlashROM content- a UFC or ADB file.

If this Tcl parameter is missing, FlashPoint tries to use the ADB as a source of FROM configuration and content data.

Values are shown in table below:

Value	Description
adb	(default)FROM content is taken from your ADB. Configurations from your UFC and ADB files are not compared.

Value	Description
ufc	FlashPoint uses FROM configuration and FROM content from the specified UFC file

`-set_io_state` *`value`*

Sets the I/O state during programming by port name or pin number. You can also use this argument to save or load an IOS file.

To set the I/O by port name, use `-set_io_state {portName:<name>; state:<state>}`. To set the I/O port by pin number, use `-set_io_state {pinNumber:<number>; state:<state>}`. To set all I/Os to the specified state, use `-set_io_state {all; state:<state>}`.

To set BSR values for an I/O, use `-set_io_state { pinNumber:<pin>; input:<state>; output_enable:<state>;output:<state> }`. See the Boundary Scan Registers - Show BSR Details section of the FlashPoint help for more information on setting Boundary Scan Registers in your device.

The following table shows the possible values for this option if you have NOT set BSR values.

Value	Description
Z	Tri-State - Sets the I/O state to tristate
Last Known State	Sets the I/O to the last known state
1	High - Sets the I/O state to high
0	Low - Sets the I/O state to low

The following table shows the possible values for this option if you have set custom BSR values.

Value	Description
Last State	Sets the I/O to the last known state
1	High - Sets the I/O state to high
0	Low - Sets the I/O state to low

To save an IOS file use the argument `-set_io_state { save:<filepath> }`

To load an IOS file, use the argument `-set_io_state { load:<filepath> }`

`-efm_block_security{location:`*`X`*`;security_level:` *`value`*`}`

This option is available only for Fusion; this argument only applies when programming the security settings (setup_security) or programming previously secured devices.

'X' identifies an Embedded Flash Memory Block instance from 0-3.

Possible values for security_level:

Value	Description
clients_jtag_protect	Enables eNVM client JTAG protection; a pass key is required for this option
write_verify_protect	The security level is medium (standard) and the Embedded Flash Memory Block cannot be read, written or verified without a Pass Key

Value	Description
write_protect	The security level is write protected. The Embedded Flash Memory Block cannot be written without a Pass Key, but it is open for reading (custom FB)
encrypted	The security level is high (standard) and uses a 128-bit AES encryption
none	The Embedded Flash Memory Block can be written and read without a Pass Key

`-efm_content {location:X;source: value}`

This option is available only for Fusion; X identifies an Embedded Flash Memory Blockinstance from 0-3. Option identifies the source file for the Embedded Flash Memory Block content, either an EFC or ADB file.

If you wish to program the entire Embedded Flash Memory Block including all its clients that were programmed in previous sessions, and use ADB content for this client, this is the only parameter you must specify. If you wish to program the entire Embedded Flash Memory Block including all its clients and use the Embedded Flash Memory Block map file (EFC) you also have to specify the –efm_block parameter.

Possible values:

Value	Description
adb	(default) Embedded Flash Memory Block content is taken from your ADB
efc	FlashPoint uses the Embedded Flash Memory Block instance configuration and content from the EFC file specified in -efm_block_parameter

`-efm_block {location:X;source: value}`

This option is available only for Fusion; X identifies an Embedded Flash Memory Block (EFMB) instance from 0-3.

Config_file specifies the location of the EFMB instance configuration file (must be an EFC file with full pathname).

`-efm_client {location:X;client:value; mem_file: value}`

This option is available only for Fusion; X identifies an EFMB instance from 0-3.

You must specify the client name and its memory content file for each client of EFMB you wish to program.

Mem_file specifies the file with the memory content for the client. If a mem_file path is specified, the memory content from this file will overwrite the client content in ADB or EFC (as defined by the -efm_content argument). If the client memory file is not specified, the client memory content from the ADB or EFC file is used instead (as defined by the -efm_content argument).

`{filename}`

Specifies the path and name of the file you are exporting.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
export -format "bts_stp"
-feature "all"
-secured_device “no”
```

```
-signature "123"
-pass_key "FB318707864EC889AE2ED8904B8EB30D"
-custom_security "no"
-fpga_security_level "write_verify_protect"
-from_security_level "write_verify_protect"
-from_config_file {.\g3_test\from.ufc}
-number_of_devices "1"
-from_progfile_type "single"
-target_programmer "specific" \
 {.\flp4.stp}
```

The following example uses the -set_io_state argument:

```
export \
-format "pdb " \
-feature "setup_security" \
-secured_device "no" \
-custom_security "no" \
-security_level "write_verify_protect" \
-security_permanent "no" \
-pass_key "012EB311B02E4C9A150B0F2BD8861CA0" \
-set_io_state { portName:AG9; state:Low} \
-set_io_state { pinNumber:AG10; state:High} \
-set_io_state { pinNumber:197; state:Tri-State} \
-set_io_state { pinNumber:198; state:Low} \
-set_io_state { pinNumber:199; state:Last Known State} \
{D:/designs/Fusion/DESIGN77}
```

The following example exports a DAT file for programming:

```
export -format "dc" -feature "prog_fpga" {./top.dat}
```

Fusion example 1:

Export soc.pdb file that includes programming data for three clients of EFM block 0. EFM block configuration file ./fus_new/nvm_simple/nvm_simple.efc and clients memory files are used for generating the programming file. Clients specified as TCL parameters must be included in EFC file.

```
export -format "pdb "
-efm_content {location:0; source:efc} \
-efm_block {location:0; config_file:{./fus_new/nvm_simple/nvm_simple.efc}} \
-efm_client {location:0; client:cfiData;
mem_file:{./fus_new/nvm_exmp/input_memfiles/ram1_block_0_ram1_R0C0.mem}} \
-efm_client {location:0; client:dataStorage;
mem_file:{./fus_new/nvm_exmp/input_memfiles/datast2_asb1_smtr_ram.hex}} \
-efm_client {location:0; client:init1;
mem_file:{./fus_new/nvm_exmp/input_memfiles/datast1_asb1_acm_rtc_ram.hex}} \
 {./soc}
```

Fusion example 2:

Export soc.stp and soc.pdb files that include programming data for EFM block 0. Information regarding block configuration, which clients to program, and their memory content is taken from ADB file.

```
export -format "pdb bts_stp"
-efm_content {location:0; source:adb} \
 {./soc}
```

Fusion example 3:

Export soc.stp and soc.pdb files that include programming data for client cfiData of EFM block 0. Other clients of block 0 are not selected to be programmed. ADB file is a source for block configuration and content; EFC is ignored.

```
export -format "pdb"
-efm_content {location:0; source:adb} \
-efm_block {location:0; config_file:{./fus_new/nvm_simple/nvm_simple.efc}} \
-efm_client {location:0; client:cfiData;} \
 {./soc}
```

See Also

Exporting files

Importing files

Tcl documentation conventions

Designer Tcl Command Reference

export (Block support)

Tcl command; exports (publishes) the Block files to a specified directory, includes any added comments.

```
export -format "block"
-export_directory {value} \
-export_name "blockname" \
-placement "value"\
-routing "value"\
-comment "value" \
-export_language "value"\
-region "value"
```

Arguments

`-export_directory {value}`

Specifies the directory name for the exported *.v, *.vhd, *.cxf and *.cdb files. Value is the path and name of the directory

`-export_name "blockname"`

Specifies the prefix of the exported *.v, *.vhd, *.cxf, and *.cdb files, where blockname is the name of the prefix.

`-placement "value"`

Exports placement information. Possible values:

Value	Description
yes	Exports the placement information. Specify "yes" only if the placer state is valid and -placement is specified as "yes."
no	Do not export the placement information.

`-routing "value"`

Exports placement information. Possible values:

Value	Description
yes	Exports routing information. Specify "yes" only if the routing state is valid and -placement is specified as "yes."

Value	Description
no	Do not export the routing information.

`-comment "value"`

Adds comments to document the block.

`-export_language "value"`

Specifies the export format of the CXF file for Libero SoC. Possible values:

Value	Description
VERILOG	CXF file is Verilog.
VHDL	CXF file is VHDL.

`-region "value"`

Option to publish all the user regions and make them available when you instantiate the block. Possible values:

Value	Description
YES	Publishes all the user regions, makes them available when you instantiate your block.
NO	Disables region publishing

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
export -format "block" -export_directory {.} -export_name "test_core" -placement "yes" -
routing "yes" -comment "toto" -export_language "VERILOG"
```

See Also

Exporting files

Importing files

Tcl documentation conventions

Designer Tcl Command Reference

generate_probes (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; executes the probing and creates a new ADB file. This command is used in conjunction with the add_probe Tcl command (see example below).

```
generate_probes –save <ADB_file_name>
```

Arguments

`-save <ADB_file_name>`

Name of the new ADB file with your probed nets.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The example below adds a probe to the net net2 on pin 4 and port prb2 with the add_probe command, and generates the new ADB file test1.adb.

```
add_probe –net net2 –pin 4 –port prb2
generate_probes –save test1.adb
```

See Also

add_probe

Generating a Probed Design

Generate Probed Design - Add Probe(s) Dialog Box

Designer Tcl Command Reference

get_cells

Tcl command; returns an object representing the cells (instances) that match those specified in the pattern argument.

```
get_cells pattern
```

Arguments

`pattern`

Specifies the pattern to match the instances to return. For example, "get_cells U18*" returns all instances starting with the characters "U18", where “*” is a wildcard that represents any character string.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

This command returns a collection of instances matching the pattern you specify. You can only use this command as part of a –from, -to, or –through argument in the following Tcl commands: set_max delay, set_multicycle_path, and set_false_path.

Examples

```
set_max_delay 2 -from [get_cells {reg*}] -to [get_ports {out}]
set_false_path –through [get_cells {Rblock/muxA}]
```

See Also

get_clocks

get_nets

get_pins

get_ports

Tcl Command Documentation Conventions

Designer Tcl Command Reference

get_clocks

Tcl command; returns an object representing the clock(s) that match those specified in the pattern argument in the current timing scenario.

```
get_clocks pattern
```

Arguments

pattern

Specifies the pattern to use to match the clocks set in SmartTime or Timer.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

- If this command is used as a –from argument in either the set maximum (set_max_delay), or set minimum delay (set_min_delay), false path (set_false_path), and multicycle constraints (set_multicycle_path), the clock pins of all the registers related to this clock are used as path start points.
- If this command is used as a –to argument in either the set maximum (set_max_delay), or set minimum delay (set_min_delay), false path (set_false_path), and multicycle constraints (set_multicycle_path), the synchronous pins of all the registers related to this clock are used as path endpoints.

Example

```
set_max_delay -from [get_ports datal] -to \
[get_clocks ck1]
```

See Also

create_clock

create_generated_clock

Tcl Command Documentation Conventions

Designer Tcl Command Reference

get_current_scenario

Tcl command; returns the name of the current timing scenario.

```
get_current_scenario
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
get_current_scenario
```

See Also

set_current_scenario

Tcl documentation conventions

Designer Tcl Command Reference

get_defvar (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; provides access to the internal variables within Designer and returns its value. This command also prints the value of the Designer variable on the Log window.

```
get_defvar variable
```

Arguments

variable

The Designer internal variable.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Example 1: Prints the design name on the log window.

```
   get_defvar “DESIGN”
set variableToGet "DESIGN"
set valueOfVariable [get_defvar $variableToGet]
puts "The value is $valueOfVariable"
```

See Also

set_defvar

Designer Tcl Command Reference

get_design_filename (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; retrieves the full qualified path of the design file. The result will be an empty string if the design has not been saved to disk. This command is equivalent to the command “get_design_info DESIGN_PATH.” This command predates get_design_info and is supported for backward-compatibility.

```
get_design_filename
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Exceptions

- The command will return an error if a design is not loaded.
- The command will return an error if arguments are passed.

Example

```
if { [ is_design_loaded ] } {
  set design_location [ get_design_filename ]
  if {$design_location != "" } {
    puts “Design is at $design_location.”
  } else {
    puts “Design has not been saved to a file on disk.”
```

```
  }
} else {
  puts "No design is loaded."
}
```

See Also

get_design_info
is_design_loaded
is_design_modified
is_design_state_complete
Designer Tcl Command Reference

get_design_info (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; retrieves some basic details of your design. The result value of the command will be a string value.

```
get_design_info value
```

Arguments

value

Must be one of the valid string values summarized in the table below:

Value	Description
name	Design name. The result is set to the design name string.
family	Silicon family. The result is set to the family name.
design_path	Fully qualified path of the design file. The result is set to the location of the .adb file. If a design has not been saved to disk, the result will be an empty string. This command replaces the command get_design_filename.
design_folder	Directory (folder) portion of the design_path.
design_file	Filename portion of the design_path.
cwdir	Current working directory. The result is set to the location of the current working directory
die	Die name. The result is set to the name of the selected die for the design. If no die is selected, this is an empty string.
Package	Package. The result is set to the name of the selected package for the design. If no package is selected, this is an empty string.
Speed	Speed grade. The result is set to the speed grade for the design. If no speed grade is selected, this is an empty string.

Supported Family

See the Tcl Commands and Supported Families table for the list of families that support this command.

Exceptions

- Returns an error if a design is not loaded.
- Returns an error if more than one argument is passed.
- Returns an error if the argument is not one of the valid values.

Example

The following example uses get_design_info to display the various values to the screen.

```
if { [ is_design_loaded ] } {
  puts "Design is loaded."
  set bDesignLoaded 1
} else {
  puts "No design is loaded."
  set bDesignLoaded 0
}
if { $bDesignLoaded != 0 } {
  set var [ get_design_info NAME ]
  puts "  DESIGN NAME:\t$var"
  set var [ get_design_info FAMILY ]
  puts "  FAMILY:\t$var"
  set var [ get_design_info DESIGN_PATH ]
  puts "  DESIGN PATH:\t$var"
  set var [ get_design_info DESIGN_FILE ]
  puts "  DESIGN FILE:\t$var"
  set var [ get_design_info DESIGN_FOLDER ]
  puts "  DESIGN FOLDER:\t$var"
  set var [ get_design_info CWDIR ]
  puts "  WORKING DIRECTORY:  $var"
  set var [ get_design_info DIE ]
  puts "  DIE:\t$var"
  set var [ get_design_info PACKAGE ]
  puts "  PACKAGE:\t'$var'"
  set var [ get_design_info SPEED ]
  puts "  SPEED GRADE:\t$var"
  if { [ is_design_modified ] } {
    puts "The design is modified."
  } else {
    puts "The design is unchanged"
  }
}
puts "get_design.tcl done"
```

See Also

get_design_filename
is_design_loaded
is_design_modified
is_design_state_complete
Designer Tcl Command Reference

get_nets

Tcl command; returns an object representing the nets that match those specified in the pattern argument.

```
get_nets pattern
```

Arguments

pattern

Specifies the pattern to match the names of the nets to return. For example, "get_nets N_255*" returns all nets starting with the characters "N_255", where “*” is a wildcard that represents any character string.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

This command returns a collection of nets matching the pattern you specify. You can only use this command as source objects in create clock (create_clock) or create generated clock (create_generated_clock) constraints and as `-through` arguments in the set false path, set minimum delay, set maximum delay, and set multicycle path constraints.

Examples

```
set_max_delay 2 -from [get_ports RDATA1] -through [get_nets {net_chkp1 net_chkqi}]
set_false_path –through [get_nets {Tblk/rm/n*}]
create_clock -name mainCLK -period 2.5 [get_nets {cknet}]
```

See Also

create_clock
create_generated_clock
set_false_path
set_min_delay
set_max_delay
set_multicycle_path
Tcl documentation conventions
Designer Tcl Command Reference

get_out_of_date_files (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; audits all files returns a list of filenames that are out of date; each filename is separated by a space. The command returns a string of file names that are out of date separated by a space i.e. file1 file2 ...

It returns empty string if all files are current.

This command ignores the Audit settings in your ADB file.

```
get_out_of_date_files
```

Arguments

None

Supported Family

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following code returns a list of filenames that are out of date.

```
get_out_of_date_files
```

See Also

are_all_source_files_curent

is_source_file_current

Designer Tcl Command Reference

get_pins

Tcl command; returns an object representing the pin(s) that match those specified in the pattern argument.

```
get_pins pattern
```

Arguments

pattern

Specifies the pattern to match the pins to return. For example, "get_pins clock_gen*" returns all pins starting with the characters "clock_gen", where “*” is a wildcard that represents any character string.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
create_clock -period 10 [get_pins clock_gen/reg2:Q]
```

See Also

create_clock

create_generated_clock

set_clock_latency

set_false_path

set_min_delay

set_max_delay

set_multicycle_path

Tcl documentation conventions

Designer Tcl Command Reference

get_ports

Tcl command; returns an object representing the port(s) that match those specified in the pattern argument.

```
get_portspattern
```

Argument

pattern

Specifies the pattern to match the ports.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
create_clock -period 10 [get_ports CK1]
```

See Also

create_clock
set_clock_latency
set_input_delay
set_output_delay
set_min_delay
set_max_delay
set_false_path
set_multicycle_path
Tcl documentation conventions
Designer Tcl Command Reference

import_aux

Tcl command; imports the specified auxiliary file into the design. Equivalent to executing the Import Auxiliary Files command from the File menu.

```
import_aux
-format file_type-partial_parse value
-start_time value
-end_time value
-auto_detect_top_level_name value
-top_level_name value
-glitch_filtering value
-glitch_threshold value
filename
```

Arguments

`-format file_type`

Specifies the file format of the file to import. You can import one of the following types of files: pdc, sdc, pin, dcf, saif, vcd, or crt.

`-partial_parse {value}`

Specifies whether to partially parse the *.vcd file. The following table shows the acceptable values for this argument:

Value	Description
true	Partially parses the *.vcd file
false	Does not partially parse the *.vcd file

`-start_time {value}`

This option is available only if `-partially_parse` is set to *true*. Specifies the start time (in ns) to partially parse the *.vcd file.

`-end_time {value}`

This option is available only if `-partially_parse` is set to *`true`*. Specifies the end time (in ns) to partially parse the *.vcd file.

`-auto_detect_top_level_name {`*`value`*`}`

Specifies whether to automatically detect the top-level name. The following table shows the acceptable values for this argument:

Value	Description
true	Automatically detects the top-level name
false	Does not automatically detect the top-level name

`-top_level_name` *`top_level_name`*

Specifies the instance name of your design in the simulation testbench when you import a VCD or SAIF file.

When importing a VCD file, the automatic *top_level_name* detection is available. If the -top_level_name option is not specified, SmartPower will try to automatically detect the top level name.

When importing a SAIF file, the automatic *top_level_name* detection is not available and -top_level_name is a required argument.

To identify the top_level_name for SAIF and VCD files manually, refer to Importing a VCD file and Importing a SAIF file.

`-glitch_filtering {`*`value`*`}`

Specifies whether to use glitch filtering. The following table shows the acceptable values for this argument:

Value	Description
true	Glitch filtering is on
auto	Enables automatic glitch filtering. This option will ignore any value specified in `-glitch_threshold`
false	Glitch filtering is off

`-glitch_threshold {`*`value`*`}`

This option is only available when `-glitch_filtering` is set to *`true`*. Specifies the glitch filtering value in ps.

`filename`

Specifies the name of the auxiliary file to import.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

- Auxiliary files are not audited and are handled as one-time data-entry or data-change events, similar to entering data using one of the interactive editors (for example, PinEditor or Timer).
- If you import the SDC file as an auxiliary file, you do not have to re-compile your design. However, auditing is disabled when you import auxiliary files, and Designer cannot detect the changes to your SDC file(s) if you import them as auxiliary files.

Examples

```
import_aux -format sdc file.sdc
```

```
import_aux -format pdc file.pdc
import_aux -format vcd file.vcd // automatic detection of top level name
import_aux -format vcd -glitch_filter 10 // filter out glitches that are 10 ps or less
import_aux -format saif -top_level_name "top" file.saif
```

See Also

import_source

Importing auxiliary files

Importing source files

Importing files

Tcl documentation conventions

Designer Tcl Command Reference

import_source

Tcl command; imports the specified source file into the design. Equivalent to executing the Import Source Filecommand from the File menu in Designer.

All source files must be specified on one command line.

```
import_source [-merge_timing value][-merge_physical value][-merge_all value][-format
file_type][-abort_on_error value][-top_entity][-edif -edif_flavor value]filename
```

Arguments

`-merge_timing` *value*

Specifies whether to preserve all existing timing constraints when you import an SDC file. Same as selecting or unselecting the "Keep existing timing constraints" check box in the Import Files dialog box. The following table shows the acceptable values for this option:

Value	Description
yes	Designer merges the timing constraints from the imported SDC file with the existing constraints saved in the constraint database. If there is a conflict, the new constraint has priority over the existing constraint.
no	All existing timing constraints are replaced by the constraints in the newly imported SDC file.

`-merge_physical` *value*

Specifies whether to preserve all existing physical constraints when you import a PDC file. Same as selecting or unselecting the "Keep existing physical constraints" check box in the Import Files dialog box. The following table shows the acceptable values for this option:

Value	Description
yes	Designer preserves all existing physical constraints that you have entered either using one of the MVN tools (ChipPlanner, PinEditor, or the I/O Attribute Editor) or a previous GCF or PDC file. The software resolves any conflicts between new and existing physical constraints and displays the appropriate message.
no	All existing physical constraints are replaced by the constraints in the newly imported GCF or PDC file.

`-merge_all value`

Specifies whether to preserve all existing physical and timing constraints when you import an SDC and/or a PDC file. Same as selecting or unselecting the "Keep existing physical constraints" and "Keep existing timing constraints" check boxes in the Import Files dialog box. The following table shows the acceptable values for this option:

Value	Description
yes	Designer preserves all existing physical constraints that you have entered either using one of the MVN tools (ChipPlanner, PinEditor, or the I/O Attribute Editor) or a previous GCF or PDC file. The software resolves any conflicts between new and existing physical constraints and displays an appropriate message. Any existing timing constraints from your ADB are merged with the new information from your imported files. New constraints override any existing timing constraints whenever there is a conflict
no	All the physical constraints in the newly imported GCF or PDC files are used. All pre-existing physical constraints are lost. Existing timing constraints from the ADB are replaced by the new timing constraints from your imported file.

`-format file_type`

Specifies the file format of the file to import. You can import one of the following types of files: adl, edif, verilog, vhdl, gcf, pdc, sdc, or crt.

Note: Refer to Importing source files to know the formats supported for each family.

`-abort_on_error value`

Aborts a PDC file if it encounters an error during import. Possible values are

Value	Description
yes	Designer aborts on error.
no	Designer ignores the error and continues.

`-top_entity`

Specifies the top entity to a VHDL file.

`-edif edif_flavor value`

Specifies the type of netlist. It can be edif, viewlogic, or mgc.

`filename`

Specifies the name of the source file to import.

Supported Families

SmartFusion, IGLOO, ProASIC3 and Fusion

Exceptions

Your script -merge options vary according to family as shown below:

- The -merge_timing, -merge_physical, and -merge_all arguments are available for IGLOO, Fusion and ProASIC3 families.
- For IGLOO, Fusion, ProASIC3:

```
import_source -merge_physical yes/no -merge_timing yes/no ...
import_source -merge_all yes/no  ...
import_source -merge yes/no  ...
```

The -merge_all and -merge options map to both -merge_physical and -merge_timing options for these families.

Examples

Consider the following sample scripts:

```
import_source  \
 -merge_physical "no" \
 -merge_timing "yes"
  -format "EDIF" -edif_flavor "GENERIC" \
 {.\designs\mydesign.edn} \
 -format "sdc" \
 {.\designs\mydesign.sdc} \
 -format "pdc" -abort_on_error "no" \
 {.\designs\mydesign.pdc}

import_source \
 -merge_physical "no" \
 -format "verilog" \
 {mydesign.v}

import_source \
  -merge_physical "no" \
  -merge_timing "no" \
  -format "vhdl" -top_entity "aclass" \
   {C:/mynetlist.vhd}

import_source \
  -merge_physical "no" \
  -merge_timing "no" \
  -format "adl" {mydesign.adl}
```

See Also

import_aux

Importing auxiliary files

Importing source files

Importing files

Tcl documentation conventions

Designer Tcl Command Reference

ioadvisor_apply_suggestion (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; applies the suggestions for the selected attribute to the selected I/O(s).

```
ioadvisor_apply_suggestion -attribute {value} -io {value}
```

Arguments

`-attribute{value}`

This specifies the attribute for which the values will be applied. The following table shows the acceptable values for this argument:

Value	Description
outdrive	Applies suggested outdrive values
slew	Applies suggested slew values

`-io {value}`

This selects the I/Os for which the suggestion will be applied. To select multiple I/Os, use `-io {value}` for each I/O.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following code applies the suggested outdrive values for two I/Os.

```
ioadvisor_apply_suggestion -attribute{outdrive} -io{nPWM_out_pad} -io{PWM_out_pad}
```

See Also

ioadvisor_commit
ioadvisor_restore
ioadvisor_restore_initial_value
ioadvisor_set_outdrive
ioadvisor_set_outputload
ioadvisor_set_slew
Designer Tcl Command Reference

ioadvisor_commit (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; saves all changes in the I/O Advisor.

```
ioadvisor_commit
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following code saves all changes in the I/O Advisor:

```
ioadvisor_commit
```

See Also

ioadvisor_apply_suggestion
ioadvisor_restore
ioadvisor_restore_initial_value
ioadvisor_set_outdrive
ioadvisor_set_outputload
ioadvisor_set_slew
Designer Tcl Command Reference

ioadvisor_restore (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; restores the I/O Advisor to the initial state. All changes not committed will be lost.

```
ioadvisor_restore
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following code restores the I/O Advisor to the initial state:

```
ioadvisor_restore
```

See Also

ioadvisor_apply_suggestion
ioadvisor_commit
ioadvisor_restore_initial_value
ioadvisor_set_outdrive
ioadvisor_set_outputload
ioadvisor_set_slew
Designer Tcl Command Reference

ioadvisor_restore_initial_value (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; sets the current value for the selected attribute and I/Os to the initial value.

```
ioadvisor_restore_initial_value -attribute {value} -io {value}
```

Arguments

`-attribute{value}`

This specifies the attribute for which the values will be restored. The following table shows the acceptable values for this argument:

Value	Description
outdrive	Restores initial outdrive values
output_load	Restores initial output load values

Value	Description
slew	Restores initial slew values

`-io {value}`

This selects the I/Os for which the initial values will be restored. To select multiple I/Os, use `-io {value}` for each I/O.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following code restores the initial outdrive values for two I/Os.

```
ioadvisor_restore_initial_value -attribute{outdrive} -io{nPWM_out_pad} -io{PWM_out_pad}
```

See Also

ioadvisor_apply_suggestion
ioadvisor_commit
ioadvisor_restore
ioadvisor_set_outdrive
ioadvisor_set_outputload
ioadvisor_set_slew
Designer Tcl Command Reference

ioadvisor_set_outdrive (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; sets the outdrive for the selected I/Os.

```
ioadvisor_set_outdrive -io {value} -outdrive {value}
```

Arguments

`-io {value}`

This selects the I/Os for which the outdrive will be set. To select multiple I/Os, use `-io {value}` for each I/O.

`-outdrive {value}`

This specifies the outdrive for the selected I/Os.The outdrive must be a positive integer value within the list of possible outdrives of the I/Os.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following code sets the outdrive for two I/Os.

```
ioadvisor_set_outdrive -io{nPWM_out_pad} -io{PWM_out_pad} -outdrive{5}
```

See Also

ioadvisor_apply_suggestion
ioadvisor_commit

ioadvisor_restore
ioadvisor_restore_initial_value
ioadvisor_set_outputload
ioadvisor_set_slew
Designer Tcl Command Reference

ioadvisor_set_outputload (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; sets the output load for the selected I/Os.

```
ioadvisor_set_outputload -io {value} -outload {value}
```

Arguments

`-io {value}`

This selects the I/Os for which the output load will be set. To select multiple I/Os, use `-io {value}` for each I/O.

`-outload {value}`

This specifies the output load for the selected I/Os.The output load must be a positive integer value.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following code sets the output load for two I/Os.

```
ioadvisor_set_outputload -io{nPWM_out_pad} -io{PWM_out_pad} -outload{5}
```

See Also

ioadvisor_apply_suggestion
ioadvisor_commit
ioadvisor_restore
ioadvisor_restore_initial_value
ioadvisor_set_outdrive
ioadvisor_set_slew
Designer Tcl Command Reference

ioadvisor_set_slew (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; sets the slew for the selected I/Os.

```
ioadvisor_set_slew -io {value} -slew {value}
```

Arguments

`-io {value}`

This selects the I/Os for which the slew will be set. To select multiple I/Os, use `-io {value}` for each I/O.

`-set_slew {value}`

This specifies the slew for the selected I/Os.The following table shows the acceptable values for this argument:

Value	Description

Value	Description
high	The slew is set to high.
low	The slew is set to low. This option is not available for all I/Os.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following code sets the slew for two I/Os.

```
ioadvisor_set_slew -io{nPWM_out_pad} -io{PWM_out_pad} -slew{high}
```

See Also

ioadvisor_apply_suggestion
ioadvisor_commit
ioadvisor_restore
ioadvisor_restore_initial_value
ioadvisor_set_outdrive
ioadvisor_set_outputload
Designer Tcl Command Reference

is_design_loaded

Tcl command; returns a Boolean value (0 for false, 1 for true) indicating if a design is loaded in the Designer software. True is returned if a design is currently loaded.

```
is_design_loaded
```

Arguments

None

Supported Family

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Some Tcl commands are valid only if a design is currently loaded in Designer. Use the ‘is_design_loaded’ command to prevent runtime errors by checking for this before invoking the commands.

Exceptions

The command will return an error if arguments are passed.

Example

The following code will determine if a design has been loaded.

```
set bDesignLoaded [ is_design_loaded ]
if { $bDesignLoaded == 0 } {
  puts “No design is loaded.”
}
```

See Also

get_design_filename
get_design_info
is_design_modified
is_design_state_complete
Designer Tcl Command Reference

is_design_modified

Tcl command; returns a Boolean value (0 for false, 1 for true) indicating if a design has been modified in the Designer software. True is returned if a design has been modified.

```
is_design_modified
```

Arguments

None

Supported Family

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Some Tcl commands are valid only if a design has been modified in Designer. Use the is_design_modified command to prevent runtime errors by checking for this before invoking the commands.

Exceptions

Returns an error if arguments are passed.

Example

The following code will determine if a design has been modified.

```
set bDesignModified [ is_design_modified ]
if { $bDesignModified == 0 } {
  puts “Design has not been modified.”
}
```

See Also

get_design_filename
get_design_info
is_design_loaded
is_design_state_complete
Designer Tcl Command Reference

is_design_state_complete

Tcl command; returns a Boolean value (0 for false, 1 for true) indicating if a specific design state is valid. True is returned if the specified design state is valid.

```
is_design_state_complete value
```

Arguments

value

Must be one of the valid string values summarized in the table below:

Value	Description
SETUP_DESIGN	The design is loaded and the family has been specified for the design
DEVICE_SELECTION	The design has completed device selection (die and package). This corresponds to having successfully called the set_device command to set the die and package
NETLIST_IMPORT	The design has imported a netlist
COMPILE	The design has completed the compile command
LAYOUT	The design has completed the layout command
BACKANNOTATE	The design has exported a post-layout timing file (e.g.SDF)
PROGRAMMING_FILES	The design has exported a programming file (e.g. AFM)

Supported Family

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Certain commands can only be used after Compile or Layout has been completed. The is_design_state_complete command allows a script to check the design state before calling one of these state-limited commands.

Exceptions

The command will return an error if a design is not loaded.

The command will return an error if more than one argument is passed.

The command will return an error if the argument is not one of the valid values.

Example

The following code runs layout, but checks that the design state for layout is complete before calling backannotate.

```
layout –timing_driven
set bLayoutDone [ is_design_state_complete LAYOUT ]
if { $bLayoutDone  != 0 } {
  backannotate -name {mydesign_ba}  -format "SDF" -language "verilog"
  }
}
```

See Also

compile

get_design_filename

`get_design_info`
`is_design_loaded`
`is_design_modified`
`layout`
`set_design`
`set_device`
Designer Tcl Command Reference

is_source_file_current (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; audits the source file and determines whether or not the file is out of date / imported into the workspace. Returns '0' if file_name is out of date or has not been imported into the workspace, and returns '1' if file_name is current.

This command ignores the Audit settings in your ADB file.

```
is_source_file_current(filename)
```

Arguments

filename is the path to the source file

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following code determines whether or not the file has been imported into the workspace.

```
is_source_file_current (./hdl/adder.vhd)
```

See Also

`are_all_source_files_curent`
`get_out_of_date_files`
Designer Tcl Command Reference

layout - SmartFusion, IGLOO, ProASIC3 and Fusion

Tcl command is identical to the layout command in the Designer GUI. Refer to the Advanced Layout Options below for more information.

```
layout
[-timing_driven | -standard]
[–power_driven value]
[-run_placer value]
[–place_incremental value]
[–run_router value]
[–route_incremental value]
```

Arguments

`-timing_driven|-standard`

Sets layout mode to be timing driven or standard (non-timing driven). The default is -timing_driven or the mode used in the previous layout command.

`-power_driven value`

The following table shows the acceptable values for this argument:

Value	Description
off	Does not run power-driven layout. This is the default.
on	Enables power-driven layout

`-place_incremental` *value*

The following table shows the acceptable values for this argument:

Value	Description
off	Discards previous placement. This is the default.
on	Sets the previous placement as the initial starting point
fix	Sets the previously placed macros' locations as "fixed" and continues to place the remaining ones

`-route_incremental` *value*

The following table shows the acceptable values for this argument:

Value	Description
off	Skips incremental mode, discards previous information. This is the default.
on	Invokes incremental routing and sets the previous routing information as the initial starting point

`-run_placer` *value*

The following table shows the acceptable values for this argument:

Value	Description
on	Invokes placement. This is the default.
off	Skips placement

`-run_router` *value*

The following table shows the acceptable values for this argument:

Value	Description
on	Invokes routing if placement is successful. This is the default.
off	Skips routing

layout - Advanced Options for SmartFusion, IGLOO, ProASIC3 and Fusion

This is equivalent to executing commands within the Advanced Layout Options dialog box.

```
[-placer_high_effort value]
[-seq_opt value]
[-mindel_repair value]
[-placer_seed value]
[-show_placer_seed]
```

Arguments

`-placer_high_effort value`

The following table shows the acceptable values for this argument:

Value	Description
off	Disables physical synthesis of combinational logic. This is the default.
on	Enables physical synthesis of combinational logic

`-seq_opt value`

The following table shows the acceptable values for this argument:

Value	Description
off	Disables physical synthesis of sequential logic. This is the default.
on	Enables physical synthesis of sequential logic in high-effort mode

`-mindel_repair value`

The following table shows the acceptable values for this argument:

Value	Description
off	Does not run minimum delay violations repair. This is the default.
on	Enables repair of minimum delay violations during route

`-placer_seed value`

An integer value that you can set to change the initial random seed number for the placement.

`-show_placer_seed value`

Causes Layout to display the initial random seed number used for the placement.

Example

```
layout
layout –place_incremental FIX –route_incremental ON
layout –placer_high_effort ON
layout –run_placer OFF –route_incremental ON –mindel_repair ON
layout –timing_driven –power_driven ON
layout –placer_seed 120
```

See Also

Place and Route (Layout)

SmartFusion, IGLOO, ProASIC3 and Fusion Advanced Place and Route (Layout) Options

Designer Tcl Command Reference

list_clocks

Tcl command; returns details about all of the clock constraints in the current timing constraint scenario.

```
list_clocks
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_clocks]
```

See Also

create_clock
remove_clock
Tcl documentation conventions
Designer Tcl Command Reference

list_clock_latencies

Tcl command; returns details about all of the clock latencies in the current timing constraint scenario.

```
list_clock_latencies
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_clock_latencies]
```

See Also

set_clock_latency
remove_clock_latency
Tcl documentation conventions
Designer Tcl Command Reference

list_clock_uncertainties

Tcl command; returns details about all of the clock uncertainties in the current timing constraint scenario.

```
list_clock_uncertainties
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
list_clock_uncertainties
```

See Also

set_clock_uncertainty
remove_clock_uncertainty
Designer Tcl Command Reference

list_disable_timings

Tcl command; returns the list of disable timing constraints for the current scenario.

```
list_disable_timings
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
list_disable_timings
```

See Also

Designer Tcl Command Reference

list_false_paths

Tcl command; returns details about all of the false paths in the current timing constraint scenario.

```
list_false_paths
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_false_paths]
```

See Also

set_false_path
remove_false_path
Tcl documentation conventions
Designer Tcl Command Reference

list_generated_clocks

Tcl command; returns details about all of the generated clock constraints in the current timing constraint scenario.

```
list_generated_clocks
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_generated_clocks]
```

See Also

create_generated_clock
remove_generated_clock
Tcl documentation conventions
Designer Tcl Command Reference

list_input_delays

Tcl command; returns details about all of the input delay constraints in the current timing constraint scenario.

```
list_input_delays
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_input_delays]
```

See Also

set_input_delay
remove_input_delay
Tcl documentation conventions
Designer Tcl Command Reference

list_max_delays

Tcl command; returns details about all of the maximum delay constraints in the current timing constraint scenario.

```
list_max_delays
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_max_delays]
```

See Also

set_max_delay
remove_max_delay
Tcl documentation conventions
Designer Tcl Command Reference

list_min_delays

Tcl command; returns details about all of the minimum delay constraints in the current timing constraint scenario.

```
list_min_delays
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_min_delays]
```

See Also

set_min_delay
remove_min_delay
Tcl documentation conventions
Designer Tcl Command Reference

list_multicycle_paths

Tcl command; returns details about all of the multicycle paths in the current timing constraint scenario.

```
list_multicycle_paths
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_multicycle_paths]
```

See Also

set_multicycle_path
remove_multicycle_path
Tcl documentation conventions
Designer Tcl Command Reference

list_objects

Tcl command; returns a list of object matching the parameter. Objects can be nets, pins, ports, clocks or instances.

```
list_objects <object>
```

Arguments

Any timing constraint parameter.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following example lists all the inputs in your design:

```
list_objects [all_inputs]
```

You can also use wildcards to filter your list, as in the following command:

```
list_objects [get_ports a*]
```

See Also

Tcl documentation conventions
Designer Tcl Command Reference

list_output_delays

Tcl command; returns details about all of the output delay constraints in the current timing constraint scenario.

```
list_output_delays
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_output_delays]
```

See Also

set_output_delay

remove_output_delay
Tcl documentation conventions
Designer Tcl Command Reference

list_paths

Tcl command; returns a list of the *n* worst paths matching the arguments. The number of paths returned can be changed using the set_options -limit_max_paths <value> command.

```
list_paths
-analysis <max | min>
-format <csv | text>
-set <name>
-clock <clock name>
-type <set_type>
-from_clock <clock name>
-to_clock <clock name>
-in_to_out
-from <port/pin pattern>
-to <port/pin pattern>
```

Arguments

`-analysis <max | min>`

Specifies whether the timing analysis is done for max-delay (setup check) or min-delay (hold check). Valid values are: max or min.

`-format < text | csv >`

Specifies the list format. It can be either text (default) or csv (comma separated values). Text format is better for display and csv format is better for parsing.

`-set <name>`

Returns a list of paths from the named set. You can either use the -set option to specify a user set by its name or use both -clock and -type to specify a set.

`-clock <clock name>`

Returns a list of paths from the specified clock domain. This option requires the -type option.

`-type <set_type>`

Specifies the type of paths to be included. It can only be used along with -clock. Valid values are:

reg_to_reg -- Paths between registers

external_setup -- Path from input ports to data pins of registers

external_hold -- Path from input ports to data pins of registers

clock_to_out -- Path from registers to output ports

reg_to_async -- Path from registers to asynchronous pins of registers

external_recovery -- Path from input ports to asynchronous pins of registers

external_removal -- Path from input ports to asynchronous pins of registers

async_to_reg -- Path from asynchronous pins to registers

`-from_clock <clock name>`

Used along with -to_clock to get the list of paths of the inter-clock domain between the two clocks.

`-to_clock <clock name>`

Used along with -from_clock to get the list of paths of the inter-clock domain between the two clocks.

`-in_to_out`

Used to get the list of path between input and output ports.

`-from <port/pin pattern>`

Filter the list of paths to those starting from ports or pins matching the pattern.

`-to <port/pin pattern>`

Filter the list of paths to those ending at ports or pins matching the pattern.

Example

The following command displays the list of register to register paths of clock domain clk1:

```
puts [ list_paths -clock clk1 -type reg_to_reg ]
```

See Also

create_set

expand_path

set_options

list_scenarios

Tcl command; returns a list of names of all of the available timing scenarios.

```
list_scenarios
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
list_scenarios
```

See Also

get_current_scenario

Tcl documentation conventions

Designer Tcl Command Reference

new_design (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; creates a new design. You need all three arguments for this command. This command will set up the Designer software for importing design source files

```
new_design -name design_name -family family_name –path  pathname–block value
```

Arguments

`-name design_name`

The name of the design. This is used as the base name for most of the files generated from Designer.

`-family family_name`

The Microsemi SoC device family for which the design is being targeted.

`-path path_name`

The physical path of the directory in which the design files will be created.

`block value`

Enables or disables Block mode.The following table shows the acceptable values for this option:

Value	Description
on	Enables Block mode
off	Disables Block mode

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Example 1: Creates a new ACT3 design with the name “test” in the current folder.

```
new_design -name "test" -family "ACT3" -path {.}
```

Example 2: These set of commands create a new design through variable substitution.

```
set desName “test
set famName “ACT3”
set path {d:/examples/test}
new_design -name $desName -family $famName -path $path
```

Example 3: Design creation and catch failures

```
if  { [catch { new_design -name $desName -family $famName -path $path }] {
              Puts “Failed to create a new design”
              # Handle Failure
} else {
              puts “New design creation successful”
              # Proceed to Import source files
}
```

See Also

close_design
open_design
save_design
set_design
Designer Tcl Command Reference

open_design (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; opens an existing design into the Designer software.

```
open_design file_name
```

Note: All previously open designs must be closed before opening a new design.

Arguments

file_name

The complete .adb file path. If the complete path is not provided, then the directory is assumed to be the current working directory.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Example 1: Opens an existing design from the file “test.adb” in the current folder.

```
open_design {test.adb}
```

Example 2: Design creation and catch failures.

```
set designFile {d:/test/my_design.adb}
if  { [catch { open_design $designFile }] {
              Puts “Failed to open design”
              # Handle Failure
} else {
              puts “Design opened successfully”
              # Proceed to further processing
}
```

See Also

close_design

new_design

save_design

Designer Tcl Command Reference

pin_assign (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; use to either assign the named pin to the specified port or assign attributes to the specified port. This command has two syntax formats. The one you use depends on what you are trying to do. The first syntax format assigns the named pin to the specified port . The second one assigns attributes to the specified port.

```
pin_assign [-nofix] -port portname -pin pin_number
pin_assign -port portname [-iostd value][-iothresh value][-outload value][-slew value][-
res_pull value]
```

Arguments

`-nofix`

Unlocks the pin assignment (by default, assignments are locked).

`-port portname`

Specifies the name of the port to which the pin is assigned.

`-pin pin_number`

Specifies the alphanumeric number of the pin to assign.

`-iostd value`

Sets the I/O standard for this pin. Choosing a standard allows the software to set other attributes such as the slew rate and output loading. If the voltage standard used with the I/O is not compatible with other I/Os in the I/O bank, then assigning an I/O standard to a port will invalidate its location and automatically unassign the I/O. The following table shows the acceptable values for the supported devices:

I/O Standards table

Use the I/O Standards table to see which I/O standards can be applied to each family:

I/O Standard	IGLOO	Fusion	ProASIC3
CMOS			
CUSTOM			
GTLP25	IGLOOe only	X	ProASIC3E and ProASIC3L only

I/O Standard	IGLOO	Fusion	ProASIC3
GTLP33	IGLOOe only	X	ProASIC3E and ProASIC3L only
GTL33	IGLOOe only	X	ProASIC3E and ProASIC3L only
GTL25	IGLOOe only	X	ProASIC3E and ProASIC3L only
HSTL1	IGLOOe only	X	ProASIC3E and ProASIC3L only
HSTLII	IGLOOe only	X	ProASIC3E and ProASIC3L only
LVCMOS33	X	X	X
LVCMOS25	IGLOOe only	X	X
LVCMOS25_50	X	X	X
LVCMOS18	X	X	X
LVCMOS15	X	X	X
LVCMOS12	X		ProASIC3L only
LVTTL	X	X	X
TTL	X	X	X
PCI	X	X	X
PCIX	X	X	X
SSTL2I	IGLOOe only	X	ProASIC3E and ProASIC3L only
SSTL2II	IGLOOe only	X	ProASIC3E and ProASIC3L only
SSTL3I	IGLOOe only	X	ProASIC3E and ProASIC3L only
SSTL3II	IGLOOe only	X	ProASIC3E and ProASIC3L only

Note: The LVDS and LVPECL I/O standards cannot be set through a script.

`-iothresh` *`value`*

Sets the compatible threshold level for inputs and outputs. The default I/O threshold is based upon the I/O standard. You can set the I/O Threshold independently of the I/O specification in the PinEditor tool by selecting **CUSTOM** in the I/O Standard cell. The following table shows the acceptable values for the supported devices:

Value	Description
CMOS	RTSX-S devices only. An advanced integrated circuit (IC) manufacturing process technology for logic and memory, characterized by high integration, low cost, low power, and high performance. CMOS logic uses a combination of p-type and n-type metal-oxide-semiconductor field effect transistors (MOSFETs) to implement logic gates and other digital circuits found in computers, telecommunications, and signal processing equipment.

Value	Description
LVTTL	(Low-Voltage TTL) A general purpose standard (EIA/JESDSA) for 3.3V applications. It uses an LVTTL input buffer and a push-pull output buffer.
PCI	A computer bus for attaching peripheral devices to a computer motherboard in a local bus. This standard supports both 33 MHz and 66 MHz PCI bus applications. It uses an LVTTL input buffer and a push-pull output buffer. With the aid of an external resistor, this I/O standard can be 5V-compliant for most families, excluding SmartFusion, IGLOO, ProASIC3 and Fusion families.

Note: The -iothresh attribute is also referred to as "Loading" in some families.

`-slew` *`value`*

Sets the output slew rate. Slew control affects only the falling edges. Rising edges are not affected. This attribute is only available for LVTTL, PCI, and PCI outputs. For LVTTL, it can either be high or low. For PCI and PCIX, it can only be set to high. The following table shows the acceptable values for the supported devices (SmartFusion, IGLOO, ProASIC3, Fusion):

Value	Description
high	Sets the I/O slew to high
low	Sets the I/O slew to low

`-res_pull` *`value`*

Allows you to include a weak resistor for either pull-up or pull-down of the input buffer. The following table shows the acceptable values for the supported devices (SmartFusion, IGLOO, ProASIC3, Fusion):

Value	Description
up	Includes a weak resistor for pull-up of the input buffer
down	Includes a weak resistor for pull-down of the input buffer
none	Does not include a weak resistor

`-out_load` *`value`*

Indicates the output-capacitance value based on the I/O standard selected. This option is not available in software. This attribute determines what Timer will use as the loading on the output pin and applies only to outputs. You can enter a capacitive load as an integral number of picofarads (*pF*). The default is 35*pF*. This attribute is available only for the following devices: SmartFusion, ProASIC3, Fusion.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

You must use `pin_commit` after the pin_assign command to save the changes to your design:

```
pin_assign -port usw0 -pin A2
pin_commit

pin_assign -port usw0 -iostd LVTTL -slew low -res_pull down
pin_commit
```

Note: To use a name with special characters such as square brackets [ ], you must put the entire name between curly braces { } or put a slash character \ immediately before each square bracket as shown in the following examples.

Note: The following example shows a port name enclosed with curly braces:

Note: The next example shows each square bracket preceded by a slash:

```
pin_assign -port LFSR_OUT\[15\] -iostd lvttl -slew High
```

See Also

pin_commit
pin_fix
pin_unassign
Tcl documentation conventions
Designer Tcl Command Reference

pin_commit

Tcl command; saves the pin assignments to the design (.adb) file.

```
pin_commit
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

To save pin assignments in your design, you must add the pin_commit command to the end of the script:

```
pin_commit
```

See Also

pin_fix
pin_unfix
pin_assign
pin_unassign
Tcl documentation conventions
Designer Tcl Command Reference

pin_fix (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; locks the pin assignment for the specified port, so the pins cannot be moved during place-and-route.

```
pin_fix -port portname
```

Arguments

`-port` *portname*

Specifies the name of the port to which the pin must be locked at its assigned location.

Note: You can assign only one pin to a port

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Fixed pins are locked pins. You cannot move locked pins during place-and-route.

Examples

You must use `pin_commit` after the pin_fix command to save the changes to your design:

```
pin_fix –port clk
pin_commit
```

See Also

`pin_commit`
`pin_unfix`
`pin_assign`
`pin_unassign`
Tcl documentation conventions
Designer Tcl Command Reference

pin_fix_all

Tcl command; locks all the assigned pins on the device so they cannot be moved during place-and-route.

```
pin_fix_all
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Fixed pins are locked pins. This command locks all the pins in your design. You cannot move locked pins during place-and-route.

Example

You must use `pin_commit` after the pin_fix_all command to save the changes to your design:

```
pin_fix_all
pin_commit
```

See Also

`pin_commit`
`pin_fix`
`pin_unfix`

pin_assign
pin_unassign
Tcl documentation conventions
Designer Tcl Command Reference

pin_unassign

Tcl command; unassigns the pin from the specified port. The unassigned pin location is then available for other ports. (Only one pin can be assigned to a port.)

```
pin_unassign -port portname
```

Arguments

`-port portname`

Specifies the name of the port for which the pin must be unassigned.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

You must use `pin_commit` after the pin_assign command to save the changes to your design:

```
pin_unassign –port “clk”
pin_commit
```

See Also

pin_commit
pin_fix
pin_fix_all
pin_unfix
pin_assign
pin_unassign
Tcl documentation conventions
Designer Tcl Command Reference

pin_unassign_all (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; unassigns all the pins from all the ports so that all pin locations are available for assignment.

```
pin_unassign_all
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

You must use `pin_commit` after the pin_assign_all command to save the changes to your design:

```
pin_unassign_all
```

```
pin_commit
```

See Also

pin_commit
pin_fix
pin_unfix
pin_assign
pin_unassign
Tcl documentation conventions
Designer Tcl Command Reference

remove_all_constraints

Tcl command; removes all timing constraints from analysis.

```
remove_all_constraints
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
remove_all_constraints
```

See Also

remove_scenario

remove_clock

Tcl command; removes the specified clock constraint from the current timing scenario.

```
remove_clock -name clock_name| -id constraint_ID
```

Arguments

`-name clock_name`

Specifies the name of the clock constraint to remove from the current scenario. You must specify either a clock name or an ID.

`-id constraint_ID`

Specifies the ID of the clock constraint to remove from the current scenario. You must specify either an ID or a clock name that exists in the current scenario.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes the specified clock constraint from the current scenario. If the specified name does not match a clock constraint in the current scenario, or if the specified ID does not refer to a clock constraint, this command fails.

Do not specify both the name and the ID.

Exceptions

You cannot use wildcards when specifying a clock name.

Examples

The following example removes the clock constraint named "my_user_clock":

```
remove_clock -name my_user_clock
```

The following example removes the clock constraint using its ID:

```
set clockId [create_clock –name my_user_clock –period 2]
remove_clock –id $clockId
```

See Also

create_clock

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_clock_latency

Tcl command; removes a clock source latency from the specified clock and from all edges of the clock.

```
remove_clock_latency {-source clock_name_or_source |-id constraint_ID}
```

Arguments

`-source` *`clock_name_or_source`*

Specifies either the clock name or source name of the clock constraint from which to remove the clock source latency. You must specify either a clock or source name or its constraint ID.

`-id` *`constraint_ID`*

Specifies the ID of the clock constraint to remove from the current scenario. You must specify either a clock or source name or its constraint ID.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes a clock source latency from the specified clock in the current scenario. If the specified source does not match a clock with a latency constraint in the current scenario, or if the specified ID does not refer to a clock with a latency constraint, this command fails.

Do not specify both the source and the ID.

Exceptions

You cannot use wildcards when specifying a clock name.

Examples

The following example removes the clock source latency from the specified clock.

```
remove_clock_latency -source my_clock
```

See Also

set_clock_latency

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_clock_uncertainty

Tcl command; removes a clock-to-clock uncertainty from the current timing scenario by specifying either its exact arguments or its ID.

```
remove_clock_uncertainty -from | -rise_from | -fall_from from_clock_list -to | -rise_to| -fall_to to_clock_list -setup {value} -hold {value}
remove_clock_uncertainty -id constraint_ID
```

Arguments

`-from`

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the source clock list. Only one of the `-from`, `-rise_from`, or `-fall_from` arguments can be specified for the constraint to be valid.

`-rise_from`

Specifies that the clock-to-clock uncertainty applies only to rising edges of the source clock list. Only one of the `-from`, `-rise_from`, or `-fall_from` arguments can be specified for the constraint to be valid.

`-fall_from`

Specifies that the clock-to-clock uncertainty applies only to falling edges of the source clock list. Only one of the `-from`, `-rise_from`, or `-fall_from` arguments can be specified for the constraint to be valid.

`from_clock_list`

Specifies the list of clock names as the uncertainty source.

`-to`

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the destination clock list. Only one of the `-to`, `-rise_to` , or `-fall_to` arguments can be specified for the constraint to be valid.

`-rise_to`

Specifies that the clock-to-clock uncertainty applies only to rising edges of the destination clock list. Only one of the `-to`, `-rise_to` , or `-fall_to` arguments can be specified for the constraint to be valid.

`-fall_to`

Specifies that the clock-to-clock uncertainty applies only to falling edges of the destination clock list. Only one of the `-to`, `-rise_to` , or `-fall_to` arguments can be specified for the constraint to be valid.

`to_clock_list`

Specifies the list of clock names as the uncertainty destination.

`-setup`

Specifies that the uncertainty applies only to setup checks. If none or both `-setup` and `-hold` are present, the uncertainty applies to both setup and hold checks.

`-hold`

Specifies that the uncertainty applies only to hold checks. If none or both `-setup` and `-hold` are present, the uncertainty applies to both setup and hold checks.

`-id` *`constraint_ID`*

Specifies the ID of the clock constraint to remove from the current scenario. You must specify either the exact parameters to set the constraint or its constraint ID.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes a clock-to-clock uncertainty from the specified clock in the current scenario. If the specified arguments do not match clocks with an uncertainty constraint in the current scenario, or if the specified ID does not refer to a clock-to-clock uncertainty constraint, this command fails.

Do not specify both the exact arguments and the ID.

Examples

```
remove_clock_uncertainty -from Clk1 -to Clk2
remove_clock_uncertainty -from Clk1 -fall_to { Clk2 Clk3 } -setup
remove_clock_uncertainty 4.3 -fall_from { Clk1 Clk2 } -rise_to *
remove_clock_uncertainty 0.1 -rise_from [ get_clocks { Clk1 Clk2 } ] -fall_to { Clk3 Clk4 } -setup
remove_clock_uncertainty 5 -rise_from Clk1 -to [ get_clocks {*} ]
remove_clock_uncertainty -id $clockId
```

See Also

remove_clock

remove_generated_clock

set_clock_uncertainty

Designer Tcl Command Reference

remove_disable_timing

Tcl command; removes a disable timing constraint by specifying its arguments, or its ID. If the arguments do not match a disable timing constraint, or if the ID does not refer to a disable timing constraint, the command fails.

```
remove_disable_timing -from value -to value name -id name
```

Arguments

`-from` *`from_port`*

Specifies the starting port. The –from and –to arguments must either both be present or both omitted for the constraint to be valid.

`-to` *`to_port`*

Specifies the ending port. The –from and –to arguments must either both be present or both omitted for the constraint to be valid.

`name`

Specifies the cell name where the disable timing constraint will be removed. It is an error to supply both a cell name and a constraint ID, as they are mutually exclusive. No wildcards are allowed when specifying a clock name, either alone or in an accessor command1.

`-id` *`name`*

Specifies the constraint name where the disable timing constraint will be removed. It is an error to supply both a cell name and a constraint ID, as they are mutually exclusive. No wildcards are allowed when specifying a clock name, either alone or in an accessor command1.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
remove_disable_timing -from port1 -to port2 -id new_constraint
```

Designer Tcl Command Reference

remove_false_path

Tcl command; removes a false path from the current timing scenario by specifying either its exact arguments or its ID.

```
remove_false_path [-from from_list] [-to to_list] [-through through_list] [-id constraint_ID]
remove_false_path -id constraint_ID
```

Arguments

`-from from_list`

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through through_list`

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

`-to to_list`

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-id constraint_ID`

Specifies the ID of the false path constraint to remove from the current scenario. You must specify either the exact false path to remove or the constraint ID that refers to the false path constraint to remove.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes a false path from the specified clock in the current scenario. If the arguments do not match a false path constraint in the current scenario, or if the specified ID does not refer to a false path constraint, this command fails.

Do not specify both the false path arguments and the constraint ID.

Exceptions

You cannot use wildcards when specifying a clock name, either alone or in an Accessor command such as get_pins or get_ports.

Examples

The following example specifies all false paths to remove:

```
remove_false_path -through U0/U1:Y
```

The following example removes the false path constraint using its id:

```
set fpId [set_false_path –from [get_clocks c*] –through {topx/reg/*} –to [get_ports out15] ]
remove_false_path –id $fpId
```

See Also

set_false_path

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_generated_clock

Tcl command; removes the specified generated clock constraint from the current scenario.

```
remove_generated_clock {-name clock_name | -id constraint_ID }
```

Arguments

`-name clock_name`

Specifies the name of the generated clock constraint to remove from the current scenario. You must specify either a clock name or an ID.

`-id constraint_ID`

Specifies the ID of the generated clock constraint to remove from the current scenario. You must specify either an ID or a clock name that exists in the current scenario.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes the specified generated clock constraint from the current scenario. If the specified name does not match a generated clock constraint in the current scenario, or if the specified ID does not refer to a generated clock constraint, this command fails.

Do not specify both the name and the ID.

Exceptions

You cannot use wildcards when specifying a generated clock name.

Examples

The following example removes the generated clock constraint named "my_user_clock":

```
remove_generated_clock -name my_user_clock
```

See Also

create_generated_clock

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_input_delay

Tcl command; removes an input delay a clock on a port by specifying both the clocks and port names or the ID of the input_delay constraint to remove.

```
remove_input_delay -clock clock_name port_pin_list
remove_input_delay -id constraint_ID
```

Arguments

`-clock clock_name`

Specifies the clock name to which the specified input delay value is assigned.

`port_pin_list`

Specifies the port names to which the specified input delay value is assigned.

`-id constraint_ID`

Specifies the ID of the clock with the input_delay value to remove from the current scenario. You must specify either both a clock name and list of port names or the input_delay constraint ID .

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes an input delay from the specified clocks and port in the current scenario. If the clocks and port names do not match an input delay constraint in the current scenario, or if the specified ID does not refer to an input delay constraint, this command fails.

Do not specify both the clock and port names and the constraint ID.

Exceptions

You cannot use wildcards when specifying a clock name, either alone or in an accessor command.

Examples

The following example removes the input delay from CLK1 on port data1:

```
remove_input_delay -clock [get_clocks CLK1] [get_ports data1]
```

See Also

set_input_delay

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_library

Tcl command; removes a VHDL library from your project.

```
remove_library
-library name
```

Arguments

`-library` *`name`*

Specifies the name of the library you wish to remove.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Remove (delete) a library called 'my_lib'.

```
remove_library –library my_lib
```

See Also

Project Manager Tcl Command Reference

add_library

rename_library

remove_max_delay

Tcl command; removes a maximum delay constraint from the current timing scenario by specifying either its exact arguments or its ID.

```
remove_max_delay [-from from_list] [-to to_list] [-through through_list]
remove_max_delay -id constraint_ID
```

Arguments

`-from from_list`

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through through_list`

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

`-to to_list`

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-id constraint_ID`

Specifies the ID of the maximum delay constraint to remove from the current scenario. You must specify either the exact maximum delay arguments to remove or the constraint ID that refers to the maximum delay constraint to remove.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes a maximum delay value from the specified clock in the current scenario. If the arguments do not match a maximum delay constraint in the current scenario, or if the specified ID does not refer to a maximum delay constraint, this command fails.

Do not specify both the maximum delay arguments and the constraint ID.

Exceptions

You cannot use wildcards when specifying a clock name, either alone or in an Accessor command.

Examples

The following example specifies a range of maximum delay constraints to remove:

```
remove_max_delay -through U0/U1:Y
```

See Also

set_max_delay

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_min_delay

Tcl command; removes a minimum delay constraint in the current timing scenario by specifying either its exact arguments or its ID.

```
remove_min_delay [-from from_list] [-to to_list] [-through through_list]
remove_min_delay -id constraint_ID
```

Arguments

`-from` *`from_list`*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through` *`through_list`*

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

`-to`*`to_list`*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-id` *`constraint_ID`*

Specifies the ID of the minimum delay constraint to remove from the current scenario. You must specify either the exact minimum delay arguments to remove or the constraint ID that refers to the minimum delay constraint to remove.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes a minimum delay value from the specified clock in the current scenario. If the arguments do not match a minimum delay constraint in the current scenario, or if the specified ID does not refer to a minimum delay constraint, this command fails.

Do not specify both the minimum delay arguments and the constraint ID.

Exceptions

You cannot use wildcards when specifying a clock name, either alone or in an accessor command.

Examples

The following example specifies a range of minimum delay constraints to remove:

```
remove_min_delay -through U0/U1:Y
```

See Also

`set_min_delay`

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_multicycle_path

Tcl command; removes a multicycle path constraint in the current timing scenario by specifying either its exact arguments or its ID.

```
remove_multicycle_path [-from from_list] [-to to_list] [-through through_list]
remove multicycle_path -id constraint_ID
```

Arguments

`-from` *`from_list`*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through` *`through_list`*

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

`-to`*`to_list`*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-id constraint_ID`

Specifies the ID of the multicycle path constraint to remove from the current scenario. You must specify either the exact multicycle path arguments to remove or the constraint ID that refers to the multicycle path constraint to remove.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes a multicycle path from the specified clock in the current scenario. If the arguments do not match a multicycle path constraint in the current scenario, or if the specified ID does not refer to a multicycle path constraint, this command fails.

Do not specify both the multicycle path arguments and the constraint ID.

Exceptions

You cannot use wildcards when specifying a clock name, either alone or in an accessor command.

Examples

The following example removes all paths between reg1 and reg2 to 3 cycles for setup check.

```
remove_multicycle_path -from [get_pins {reg1}] –to [get_pins {reg2}]
```

See Also

set_multicycle_path

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_output_delay

Tcl command; removes an ouput delay by specifying both the clocks and port names or the ID of the output_delay constraint to remove.

```
remove_output_delay -clock clock_name port_pin_list
remove_output_delay -id constraint_ID
```

Arguments

`-clock clock_name`

Specifies the clock name to which the specified output delay value is assigned.

`port_pin_list`

Specifies the port names to which the specified output delay value is assigned.

`-id constraint_ID`

Specifies the ID of the clock with the output_delay value to remove from the current scenario. You must specify either both a clock name and list of port names or the output_delay constraint ID .

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes an output delay from the specified clocks and port in the current scenario. If the clocks and port names do not match an output delay constraint in the current scenario, or if the specified ID does not refer to an output delay constraint, this command fails.

Do not specify both the clock and port names and the constraint ID.

Exceptions

You cannot use wildcards when specifying a clock name, either alone or in an accessor command.

Examples

The following example removes the output delay from CLK1 on port out1:

```
remove_output_delay -clock [get_clocks CLK1] [get_ports out1]
```

See Also

set_output_delay

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_scenario

Tcl command; removes a scenario from the constraint database.

```
remove_scenario <name>
```

Arguments

name

Specifies the name of the scenario to delete.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following command removes the scenario named my_scenario:

```
remove_scenario my_scenario
```

See Also

create_scenario

remove_set

Tcl command; removes a set of paths from analysis. Only user-created sets can be deleted.

```
remove_set -name name
```

Parameters

-name *name*

Specifies the name of the set to delete.

Example

The following command removes the set named my_set:

```
remove_set -name my_set
```

See Also

`create_set`

rename_scenario

Tcl command; renames the specified timing scenario with the new name provided. You must provide a unique new name (that is, it cannot already be used by another timing scenario).

```
rename_scenario oldname -new newname
```

Arguments

oldname

Specifies the current name of the timing scenario.

-new *newname*

Specifies the new name to give to the timing scenario.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

This command changes the name of the timing scenario in the list of scenarios.

Example

```
rename_scenario scenario_A -new scenario_B
```

See Also

`create_scenario`

`delete_scenario`

Tcl documentation conventions

Designer Tcl Command Reference

report

The report command provides you with frequently-used information in a convenient format.

You can generate several different types of reports using this command, including:

- report (Status)
- report (Timing) for SmartFusion2, SmartFusion, IGLOO, ProASIC3, Fusion families
- report (Timing violations) for SmartFusion2, SmartFusion, IGLOO, ProASIC3, Fusion families
- report (Pin)
- report (Flip-flop)
- report (I/O Bank)
- report (Global Usage)
- report (Power)

report (Activity and Hazards Power Report)

Tcl command; the activity and hazards report reads a VCD file and reports transitions and hazards for each clock cycle of the VCD file.

```
report -type power_activity_map \
[-vcd_file {path}] \
[-style {value}] \
[-partial_parse {value}] \
[-start_time {value}] \
[-end_time {value}] \
[-auto_detect_top_level_name {value}] \
[-top_level_name {name}] \
[-report_type {value}] \
[-report_query {value}] \
[-sortby {value }] \
[-sortorder {value }] \
[-max_net {value}] \
[-max_cycle {value}] \
[-clock_settings {value}] \
[-glitch_filtering {value}] \
[-glitch_threshold {value}] \
[-auto_construct_clock_domain {value}] \
[-clock_period {value}] \
[-clock_offset {value}] \
[-opmode {value}] \
{filename}
```

Arguments

`type -power_activity_map`

Specifies the type of report to generate is an activity and hazards power report.

`-vcd_file {path}`

Specifies the path to the *.vcd file that you want to import.

`-style {value}`

Specifies the format in which the report will be exported. The following table shows the acceptable values for this argument:

Value	Description
Text	The report will be exported as Text file
CSV	The report will be exported as CSV file

`-partial_parse {value}`

Specifies whether to partially parse the *.vcd file. The following table shows the acceptable values for this argument:

Value	Description
true	Partially parses the *.vcd file
false	Does not partially parse the *.vcd file

`-start_time {value}`

This option is available only if `-partially_parse` is set to *`true`*. Specifies the start time (in ns) to partially parse the *.vcd file.

`-end_time {`*`value`*`}`

This option is available only if `-partially_parse` is set to *`true`*. Specifies the end time (in ns) to partially parse the *.vcd file.

`-auto_detect_top_level_name {`*`value`*`}`

Specifies whether to automatically detect the top-level name. The following table shows the acceptable values for this argument:

Value	Description
true	Automatically detects the top-level name
false	Does not automatically detect the top-level name

`-top_level_name {`*`name`*`}`

Specifies the top-level name.

`-report_type {`*`value`*`}`

Specifies the report query type. The following table shows the acceptable values for this argument:

Value	Description
activity	Includes activity information for each net
power	Includes power information for each net
activity and power	Includes activity and power information for each net

`-report_query {`*`value`*`}`

Specifies the report type. The following table shows the acceptable values for this argument:

Value	Description
Report by Net - summary	Provides a summary report for each net
Report by Net - detailed	Provides a detailed report for each net
Report by Cycle - summary	Provides a summary report for each cycle
Report by cycle - detailed	Provides a detailed report for each cycle

`-sortby {`*`value`*`}`

Specifies how to sort the values in the report. The following table shows the acceptable values for this argument:

Value	Description
total power	Sorts based on the power values
spurious power	Sorts based on the spurious power
functional power	Sorts based on the functional power

Value	Description
spurious transitions	Sorts based on the spurious transitions
functional transitions	Sorts based on the functional transitions

`-sortorder {value}`

Specifies the sort order of the values in the report. This could be descending or ascending.

`-max_net {value}`

Specifies the maximum number of nets to report. In a net summary or net details report, this argument limits the total number of entries. In a cycle details report, this argument limits the number of nets reported for each cycle.

`-max_cycle {value}`

Specifies the maximum number of cycles to report. In a cycle summary or cycle details report, this argument limits the total number of entries. In a net details report, this argument limits the number of cycles reported for each net

`-clock_settings {value}`

Specifies the settings for the clock. The format is *"< clock name >:< active edge { value } >"*. The following table shows the acceptable values for the active edge:

Value	Description
rising	Sets the clock to a rising active edge
falling	Sets the clock to a falling active edge
both	Sets the clock to both rising and falling active edge
not_active	Does not use the signal as a clock

`-glitch_filtering {value}`

Specifies whether to use glitch filtering. The following table shows the acceptable values for this argument:

Value	Description
true	Glitch filtering is on
auto	Enables automatic glitch filtering. This option will ignore any value specified in `-glitch_threshold`
false	Glitch filtering is off

`-glitch_threshold {value}`

This option is only available when `-glitch_filtering` is set to *true*. Specifies the glitch filtering value in ps.

`-auto_construct_clock_domain {value}`

Specifies whether to automatically construct the clock domain. The following table shows the acceptable values for this argument:

Value	Description
true	Automatically constructs the clock domain

Value	Description
false	Does not automatically construct the clock domain

`-clock_period {value}`

Use this option to specify a virtual clock period (in ps). This should be used if `-auto_construct_clock_domain` is set to *false*.

`-clock_offset {value}`

Use this option to specify the time of the first active edge of the virtual clock (in ps). This should be used if `–auto_construct_clock_domain` is set to *false*.

`-opmode {value}`

Use this option to specify the mode from which the operating conditions are extracted to generate the report.

Value	Description
Active	The operating mode is set to Active
Standby	The operating mode is set to Standby
Flash*Freeze	The operating mode is set to Flash*Freeze

`{filename}`

Specifies the name of the report.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example generates an activity and hazards power report named *report_power_activity_map.txt*.

```
report -type "power_activity_map" -vcd_file "D:/FPU/mul.vcd" -style "Text" -
partial_parse "TRUE" -start_time "0.05" -end_time "1.00" -auto_detect_top_level_name
"TRUE" -report_query "Report by Net - Summary" -clock_settings
"UUT/un3_out_3:Y:not_active" -clock_settings "clk:rising" -glitch_filtering "FALSE" -
glitch_threshold "100" -auto_construct_virtual_clock "TRUE" -virtual_clock_period
"10000.00" -virtual_clock_first_edge "0.00" -opmode "Active"\
{D:/FPU/report_power_activity_map.txt}
```

See Also

Designer Tcl Command Reference

report (Bottleneck) using SmartTime

Tcl command; creates a bottleneck report.

```
report -type bottleneck
 [-cost_type {value} ]
 [-use_slack_threshold{value} ]
 [-slack_threshold {value} ]
 [-set_name {value} ]
 [-clock clock_id -set_type value ]
 [-source_clock clock_id -sink_clock clock_id]
 [-source {pin_list} ]
```

```
[-sink {pin_list} ]
[-max_instances {value} ]
[-max_paths {value} ]
[-max_parallel_paths {value} ]
[-analysis {value} ]
{filename} \
[-format value]
```

Arguments

`-cost_type value`

Specifies the type of bottleneck cost. The default option is path_count.

Value	Description
path_count	Instances with the greatest number of path violations will have the highest bottleneck cost
path_cost	Instances with the largest combined timing violations will have the highest bottleneck cost

`-use_slack_threshold value`

Specifies whether to consider the slack threshold when computing the bottlenecks in the report.

Value	Description
yes	Includes slack threshold in the bottleneck report
no	Excludes slack threshold in the bottleneck report

`-slack_threshold value`

Specifies that paths whose slack is larger than this given threshold will be considered. Only instances that lie on these violating paths are reported. The default option is 0.

`-set_name value`

Displays the bottleneck information for the named set. You can either use this option or use both –clock and –type. This option allows pruning based on a given set. Only paths that lie within the named set will be considered towards bottleneck.

`-clock value`

This option allows pruning based on a given clock domain. Only instances that lie on these violating paths are reported.

`-set_type value`

This option can only be used in combination with the –clock option, and not by itself. The options allow to filter which type of paths should be considered towards the bottleneck.

Value	Description
reg_to_reg	Paths between registers in the design
async_to_reg	Paths from asynchronous pins to registers
reg_to_async	Paths from registers to asynchronous pins
external_recovery	The set of paths from inputs to asynchronous pins
external_removal	The set of paths from inputs to asynchronous pins

Value	Description
external_setup	Paths from input ports to registers
external_hold	Paths from input ports to registers
clock_to_out	Paths from registers to output ports

`-source_clock` *`clock_id`*

Reports only bottleneck instances that lie on violating timing paths of the inter-clock domain that starts at the source clock specified by this option. This option can only be used in combination with -sink_clock, and not by itself.

`-sink_clock` *`clock_id`*

Reports only bottleneck instances that lie on violating timing paths of the inter-clock domain that ends at the sink clock specified by this option. This option can only be used in combination with -source_clock, and not by itself.

`-source` *`value`*

Reports only instances that lie on violating paths that start at locations specified by this option.

`-sink` *`value`*

Reports only instances that lie on violating paths that end at locations specified by this option.

`-max_instances` *`value`*

Specifies the maximum number of instances to be reported. Defaults to 10.

`-max_paths` *`value`*

Specifies the maximum number of paths to be considered per path set type. Allowed values are 1 to 2000000. Defaults to 100.

`-max_parallel_paths` *`value`*

Specifies the maximum number of paths allowed per end point pair. Only instances that lie on these violating paths are reported. Defaults to 1 (No parallel paths).

`-analysis` *`value`*

Specifies the analysis types (max or min) under which the violations are reported. Defaults to max analysis.

Value	Description
max	Sets the analysis type to maximum delay
min	Sets the analysis type to minimum delay

`-format` *`value`*

Specifies the output format of the generated report.

Value	Description
text	Generates a text report; text is the default value
csv	Generates the report in a comma-separated value format that you can import into a spreadsheet

`filename`

Specifies the name and destination of the bottleneck report.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

The following example generates a bottleneck report named bottleneck.txt.

```
report -type bottleneck -cost_type path_count -slack_threshold 0 -set_name set1 -
max_paths 10 -max_parallel_paths 10 -analysis max -format text bottleneck.txt
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

report (Data Change History)

Tcl command; creates a Data Change History report, which lists new features and enhancements, bug fixes and known issues for the current release that may impact the power consumption of the design.

```
report -type power_history \
{report.txt}
```

Arguments

`-type power_history`

Specifies the type of report to generate is a data change history report.

`{report.txt}`

Specifies the name of the report. You must use .txt as the filename extension.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example generates a data change history report named *report.txt*.

```
report -type "power_history" \
{report.txt}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

report (Cycle Accurate Power Report)

Tcl command; creates a cycle accurate power report, which reports a power waveform with one power value per clock period or half-period instead of an average power for the whole simulation.

```
report -type power_cycle_accurate\
[-vcd_file {path}] \
[-style {value}] \
[-partial_parse {value}] \
[-start_time {value}] \
[-end_time {value}] \
[-auto_detect_top_level_name {value}] \
[-top_level_name {name}] \
[-glitch_filtering {value}] \
```

```
[-glitch_threshold {value}] \
[-auto_detect_sampling_period {value}] \
[-sampling_clock { }] \
[-sampling_rate_per_period {value}] \
[-sampling_offset {value}] \
[-sampling_period {value}] \
[-use_only_local_extrema {value}] \
[-use_power_threshold {value}] \
[-power_threshold {value}] \
[-opmode {value}] \
{filename}
```

Arguments

`-type power_cycle_accurate`

Specifies the type of report to generate is a cycle accurate power report.

`-vcd_file {path}`

Specifies the path to the *.vcd file that you want to import.

`-style {value}`

Specifies the format in which the report will be exported. The following table shows the acceptable values for this argument:

Value	Description
Text	The report will be exported as Text file
CSV	The report will be exported as CSV file

`-partial_parse {value}`

Specifies whether to partially parse the *.vcd file. The following table shows the acceptable values for this argument:

Value	Description
true	Partially parses the *.vcd file
false	Does not partially parse the *.vcd file

`-start_time {value}`

This option is available only if `-partially_parse` is set to *true*. Specifies the start time (in ns) to partially parse the *.vcd file.

`-end_time {value}`

This option is available only if `-partially_parse` is set to *true*. Specifies the end time (in ns) to partially parse the *.vcd file.

`-auto_detect_top_level_name {value}`

Specifies whether to automatically detect the top-level name. The following table shows the acceptable values for this argument:

Value	Description
true	Automatically detects the top-level name
false	Does not automatically detect the top-level name

`-top_level_name {name}`

Specifies the top-level name.

`-glitch_filtering {value}`

Specifies whether to use glitch filtering. The following table shows the acceptable values for this argument:

Value	Description
true	Glitch filtering is on
auto	Enables automatic glitch filtering. This option will ignore any value specified in `-glitch_threshold`
false	Glitch filtering is off

`-glitch_threshold {value}`

This option is only available when `-glitch_filtering` is set to *true*. Specifies the glitch filtering value (in ps).

`-power_summary {value}`

Specifies whether to include the power summary, which shows the static and dynamic values in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the power summary in the report
false	Does not include the power summary in the report

`-auto_detect_sampling_period {value}`

Specifies whether to automatically detect the sampling period. The following table shows the acceptable values for this argument:

Value	Description
true	Automatically detects the sampling period
false	Does not automatically detect the sampling period

`-sampling_clock {}`

Specifies the sampling clock.

`-sampling_rate_per_period {value}`

Specifies whether to set the sampling rate per period. The following table shows the acceptable values for this argument:

Value	Description
true	Specifies the sampling rate per period
false	Specifies the sampling rate per half period

`-sampling_offset {value}`

Specifies the offset used to calculate the sampling offset (in ps).

`-sampling_period {value}`

Specifies the offset used to calculate the sampling period (in ps).

`-use_only_local_extrema {value}`

Specifies whether to limit the history size by keeping only local extrema. The following table shows the acceptable values for this argument:

Value	Description
true	Limits the history size by keeping only local extrema
false	Does not limit the history size by keeping only local extrema

`-use_power_threshold {value}`

Specifies whether to limit the history size by setting a power threshold. The following table shows the acceptable values for this argument:

Value	Description
true	Limits the history size by setting a power threshold
false	Does not limit the history size by setting a power threshold

`-power_threshold {value}`

Sets the power threshold value.

`-opmode {value}`

Use this option to specify the mode from which the operating conditions are extracted to generate the report.

Value	Description
Active	The operating mode is set to Active
Standby	The operating mode is set to Standby
Flash*Freeze	The operating mode is set to Flash*Freeze

`{filename}`

Specifies the name of the report.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example generates a cycle accurate power report named report_power_cycle_based.txt.

```
report -type "power_cycle_accurate" -vcd_file "D:/FPU/mul.vcd" -style "Text" -
partial_parse "TRUE" -start_time "0.05" -end_time "1.00" -auto_detect_top_level_name
"TRUE" -glitch_filtering "FALSE" -glitch_threshold "100" -auto_detect_sampling_period
"TRUE" -sampling_clock "clk" -sampling_rate_per_period "TRUE" -sampling_offset "0.00" -
sampling_period "10000.00" -use_only_local_extrema "TRUE" -use_power_threshold "TRUE" -
power_threshold "0.00" -opmode "Active" \ {D:/FPU/report_power_cycle_based.txt}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

report (Datasheet) using SmartTime

Tcl command; creates a datasheet report.

```
report -type datasheet filename \
[-format value]
```

Arguments

`filename`

Specifies the name and destination of the datasheet report.

`-format value`

Specifies the output format of the generated the report.

Value	Description
text	Generates a text report; text is the default value
csv	Generates the report in a comma-separated value format which you can import into a spreadsheet

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

The following example generates a datasheet report named datasheet.txt.

```
report -type datasheet -format Text datasheet.txt
```

See Also

Tcl documentation conventions

report (Timing) using SmartTime

report (Timing violations) using SmartTime

Designer Tcl Command Reference

report (Power)

Tcl command; creates a Power report, which enables you to determine if you have any power consumption problems in your design. It includes information about the global device and SmartPower preferences selection, and hierarchical detail (including gates, blocks, and nets), with a block-by-block, gate-by-gate, and net-by-net power summary SmartPower results.

```
report -type power \
[-powerunit {value}] \
[-frequnit {value}] \
[-opcond {value}] \
[-opmode {value}] \
[-toggle {value}] \
[-power_summary {value}] \
[-rail_breakdown{value}] \
[-type_breakdown{ value}] \
[-clock_breakdown{value}] \
```

```
[-thermal_summary {value}] \
[-battery_life {value}] \
[-opcond_summary {value}] \
[-clock_summary {value}] \
[-style {value}] \
[-sortorder {value}] \
[-sortby {value}] \
[-instance_breakdown {value}] \
[-power_threshold {value}] \
[-filter_instance {value}] \
[-min_power {number}] \
[-max_instance {integer >= 0}] \
[-activity_sortorder {value}] \
[-activity_sortby {value}] \
[-activity_summary {value}] \
[-frequency_threshold {value}] \
[-filter_pin {value}] \
[-min_frequency {value}] \
[-max_pin {value}] \
[-enablerates_sortorder {value}] \
[-enablerates_sortby {value}] \
[-enablerates_summary {value}] \
[-with_annotation_coverage {value}] \
{filename}
```

Arguments

`-type power`

Specifies the type of report to generate is a Power report.

`-powerunit {value}`

Specifies the unit in which power is set. The following table shows the acceptable values for this argument:

Value	Description
W	The power unit is set to watts
mW	The power unit is set to milliwatts
uW	The power unit is set to microwatts

`-frequnit {value}`

Specifies the unit in which frequency is set. The following table shows the acceptable values for this argument:

Value	Description
Hz	The frequency unit is set to hertz
kHz	The frequency unit is set to kilohertz
MHz	The frequency unit is set to megahertz

`-opcond {value}`

Specifies the operating condition. The following table shows the acceptable values for this argument:

Value	Description
worst	The operating condition is set to worst case
typical	The operating condition is set to typical case
best	The operating condition is set to best case

`-opmode {value}`

Specifies the operating mode. The following table shows the acceptable values for this argument:

Value	Description
Active	The operating mode is set to Active
Standby	The operating mode is set to Standby
Flash*Freeze	The operating mode is set to Flash*Freeze

`-toggle {value}`

Specifies the toggle. The following table shows the acceptable values for this argument:

Value	Description
true	The toggle is set to true
false	The toggle is set to false

`-power_summary {value}`

Specifies whether to include the power summary, which shows the static and dynamic values in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the power summary in the report
false	Does not include the power summary in the report

`-rail_breakdown {value}`

Specifies whether to include the breakdown by rail summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the breakdown by rail summary in the report
false	Does not include the breakdown by rail summary in the report

`-type_breakdown {value}`

Specifies whether to include the breakdown by type summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the breakdown by type summary in the report
false	Does not include the breakdown by type summary in the report

`-clock_breakdown {value}`

Specifies whether to include the breakdown by clock domain in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the breakdown by clock domain summary in the report
false	Does not include the breakdown by clock domain summary in the report

`-thermal_summary {value}`

Specifies whether to include the thermal summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the thermal summary in the report
false	Does not include the thermal summary in the report

`-battery_life {value}`

Specifies whether to include the battery life summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the battery life summary in the report
false	Does not include the battery life summary in the report

`-opcond_summary {value}`

Specifies whether to include the operating conditions summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the operating conditions summary in the report
false	Does not include the operating conditions summary in the report

`-clock_summary {value}`

Specifies whether to include the clock domains summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the clock summary in the report
false	Does not include the clock summary in the report

`-style {value}`

Specifies the format in which the report will be exported. The following table shows the acceptable values for this argument:

Value	Description
Text	The report will be exported as Text file
CSV	The report will be exported as CSV file

`-sortby {value}`

Specifies how to sort the values in the report. The following table shows the acceptable values for this argument:

Value	Description
power values	Sorts based on the power values
alphabetical	Sorts in an alphabetical order

`-sortorder {value}`

Specifies the sort order of the values in the report. The following table shows the acceptable values for this argument:

Value	Description
ascending	Sorts the values in ascending order
descending	Sorts the values in descending order

`-instance_breakdown {value}`

Specifies whether to include the breakdown by instance in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the breakdown by instance in the report
false	Does not include the breakdown by instance in the report

`-power_threshold {value}`

This specifies whether to include only the instances that consume power above a certain minimum value. When this command is set to true, the `-min_power` argument must also be used to specify that only the instances that consume power above this minimum power value are the ones that are included in the report. The following table shows the acceptable values for this argument:

Value	Description

Value	Description
true	Includes the power threshold in the report
false	Does not include the power threshold in the report

`-filter_instance {value}`

This specifies whether to have a limit on the number of instances to include in the Power report. When this command is set to true, the `–max_instance` argument must also be used to specify the maximum number of instances to be included into the Power report. The following table shows the acceptable values for this argument:

Value	Description
true	Indicates that you want to have a limit on the number of instances to include in the Power report
false	Indicates that you do not want to have a limit on the number of instances to include in the Power report

`-min_power {number}`

Specifies which block to expand based on the minimum power value of a block.

`-max_instance {integer >= 0}`

Sets the maximum number of instances to a specified integer greater than or equal to 0 (zero). This will limit the maximum number of instances to be included in the Power report.

`-activity_sortorder {value}`

Specifies the sort order for the activity summary. The following table shows the acceptable values for this argument:

Value	Description
ascending	Sorts the values in ascending order
descending	Sorts the values in descending order

`-activity_sortby {value}`

Specifies how to sort the values for the activity summary. The following table shows the acceptable values for this argument:

Value	Description
pin name	Sorts based on the pin name
net name	Sorts based on thepnet name
domain	Sorts based on the clock domain
frequency	Sorts based on the clock frequency
source	Sorts based on the clock frequency source

`-activity_summary {value}`

Specifies whether to include the activity summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the activity summary in the report
false	Does not include the activity summary in the report

`-frequency_threshold {value}`

Specifies whether to add a frequency threshold. The following table shows the acceptable values for this argument:

Value	Description
true	Adds a frequency threshold
false	Does not add a frequency threshold

`-filter_pin {value}`

Specifies whether to filter by maximum number of pins. The following table shows the acceptable values for this argument:

Value	Description
true	Filters by maximum number of pins
false	Des not filter by maximum number of pins

`-min_frequency {value}`

Sets the minimum frequency to {decimal value [unit { Hz | KHz | MHz}]}.

`-max_pin {value}`

Sets the maximum number of pins.

`-enablerates_sortorder {value}`

Specifies the sort order for the probabilities summary. The following table shows the acceptable values for this argument:

Value	Description
ascending	Sorts the values in ascending order
descending	Sorts the values in descending order

`-enablerates_sortby {value}`

Specifies how to sort the values for the probabilities summary. The following table shows the acceptable values for this argument:

Value	Description
pin name	Sorts based on the pin name
net name	Sorts based on the net name
domain	Sorts based on the clock domain

Value	Description
frequency	Sorts based on the clock frequency
source	Sorts based on the clock frequency source

`-enablerates_summary {value}`

Specifies whether to include the probabilities summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the activity summary in the report
false	Does not include the activity summary in the report

`-with_annotation_coverage {value}`

Specifies whether to include the annotation coverage summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the annotation coverage summary in the report
false	Does not include the annotation coverage summary in the report

`{filename}`

Specifies the name of the report.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

- The following arguments have been removed. Running the script will trigger a warning message: Warning: Invalid argument: -argname "argvalue" Ignored. Ignore the warning.

```
-annotated_pins {value}
-stat_pow {value}
-dyn_pow {value}
```

- Flash*Freeze, Sleep, and Shutdown are available only for certain families and devices.
- Worst and Best are available only for certain families and devices.

Examples

This example generates a Power report named report.rpt.

```
report -type "power" \
 "Power Values" \
 "Descending" \
"TRUE" \
 "FALSE" \
 "FALSE" \
```

```
"Typical" \
-min_power "2 mW" \
"ACTIVE" \
"TRUE" \
"TRUE" \
"TRUE" \
"TRUE" \
"5" \
"TRUE" \
{e:\SmartPower\report.rpt}
```

See Also

Designer Tcl Command Reference

report (Power Scenario)

Tcl command; creates a scenario power report for a previously defined scenario. It includes information about the global device and SmartPower preferences selection, and the average power consumption and the excepted battery life for this sequence.

```
report -type power_scenario \
[-powerunit {value}] \
[-frequnit {value}] \
[-opcond {value}] \
[-toggle {value}] \
[-scenario {value}] \
[-style {value}] \
[-battery_life {value}] \
[-battery_capacity {value}] \
[-rail_breakdown {value}] \
[-type_breakdown {value}] \
[-mode_breakdown {value}] \
[-opcond_summary {value}] \
{filename}
```

Arguments

`-type power_scenario`

Specifies the type of report to generate is a scenario power report.

`-powerunit {value}`

Specifies the unit in which power is set. The following table shows the acceptable values for this argument:

Value	Description
W	The power unit is set to watts
mW	The power unit is set to milliwatts
uW	The power unit is set to microwatts

`-frequnit {value}`

Specifies the unit in which frequency is set. The following table shows the acceptable values for this argument:

Value	Description
Hz	The frequency unit is set to hertz
kHz	The frequency unit is set to kilohertz
MHz	The frequency unit is set to megahertz

`-toggle {value}`

Specifies the toggle. The following table shows the acceptable values for this argument:

Value	Description
true	The toggle is set to true
false	The toggle is set to false

`-scenario{value}`

Specifies a scenario that the report is generated from.

`-style {value}`

Specifies the format in which the report will be exported. The following table shows the acceptable values for this argument:

Value	Description
Text	The report will be exported as Text file
CSV	The report will be exported as CSV file

`-battery_life {value}`

Specifies whether to include the battery life summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the battery life summary in the report
false	Does not include the battery life summary in the report

`-battery_capacity {value}`

Specifies the battery capacity in A*H.

`-rail_breakdown {value}`

Specifies whether to include the breakdown by rail summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the breakdown by rail summary in the report
false	Does not include the breakdown by rail summary in the report. This is the default value.

`-type_breakdown {value}`

Specifies whether to include the breakdown by type summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the breakdown by type summary in the report
false	Does not include the breakdown by type summary in the report. This is the default value.

`-mode_breakdown {value}`

Specifies whether to include a breakdown by mode in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the breakdown by mode in the report
false	Does not include the breakdown by mode in the report. This is the default value.

`-opcond_summary {value}`

Specifies whether to include the operating conditions summary in the report. The following table shows the acceptable values for this argument:

Value	Description
true	Includes the operating conditions summary in the report
false	Does not include the operating conditions summary in the report

`{filename.rpt}`

Specifies the name of the report.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

- Flash*Freeze, Sleep, and Shutdown are available only for certain families and devices.
- Worst and Best are available only for certain families and devices.

Examples

This example generates a scenario power report named report.txt for my_scenario

```
report -type power_scenario -scenario my_scenario -rail_breakdown true -type_breakdown
true -mode_breakdown true -style text -battery_capacity 10 report.txt
```

See Also

Scenario Power Report

report (Timing) using SmartTime

Tcl command; creates a timing report.

```
report -type timing \
[-print_summary value]\
[-analysis value]\
[-use_slack_threshold value]\
[-slack_threshold value]\
[-print_paths value]\
[-max_paths value]\
[-max_expanded_paths value]\
[-include_user_sets value]\
[-include_pin_to_pin value]\
[-include_clock_domains value]\
[-select_clock_domains value]\
[-clock_domain clock_domain_list]
[-format value]
filename
```

Arguments

`-type timing`

Specifies the type of report to generate.

`-print_summary` *value*

Specifies whether to print the summary section in the timing report.

Value	Description
yes	Includes summary section in the timing report (the default value).
no	Excludes summary section in the timing report

`-analysis` *value*

Specifies whether the report will consider minimum analysis or maximum analysis.

Value	Description
min	Timing report considers minimum analysis
max	Timing report considers maximum analysis (the default value)

`-use_slack_threshold` *value*

Specifies whether the report will consider slack threshold.

Value	Description
yes	Includes slack threshold in the timing report.
no	Excludes slack threshold in the timing report (the default value)

`-slack_threshold` *value*

Specifies the threshold to consider when reporting path slacks. This is a floating-point number in nanoseconds (ns). By default, there is no threshold (all slacks are reported).

`-print_paths` *value*

Specifies whether the path section (clock domains and in-to-out paths) will be printed in the timing report.

Value	Description
yes	Includes path section in the timing report (the default value)
no	Excludes path sections from the timing report

`-max_paths` *value*

Defines the maximum number of paths to display for each set. This is a positive integer value greater than zero. The default is 5.

`-max_expanded_paths` *value*

Defines the number of paths to expand per set. This is a positive integer value greater than zero. The default is 1.

`-include_user_sets` *value*

Defines whether to include the user defined sets in the timing report.

Value	Description
yes	Includes user defined sets in the timing report (the default value)
no	Excludes user defined sets from the timing report

`-include_pin_to_pin` *value*

Specifies whether to show pin-to-pin paths in the timing report.

Value	Description
yes	Includes pin-to-pin paths in the timing report (the default value).
no	Excludes pin-to-pin paths from the timing report

`-include_clock_domains` *value*

Defines whether to include clock domains in the timing report.

Value	Description
yes	Includes clock domains
no	Excludes clock domains from the timing report

`-select_clock_domains` *value*

Specifies whether to show the clock domain list in the timing report.

Value	Description
yes	Includes the clock domain list in the timing report

Value	Description
no	Excludes the clock domain list from the timing report (the default value)

`-clock_domain clock_domain_list`

Defines the clock domain to be considered in the clock domain section. The domain list is a series of strings with domain names separated by spaces. Both the summary and the path sections in the timing report display only the listed clock domains.

`-format` *value*

Specifies the output format of the generated the report.

Value	Description
text	Generates a text report; text is the default value
csv	Generates the report in a comma-separated value format which you can import into a spreadsheet

filename

Specifies the name and destination of the timing report.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

The following example generates a timing report named timing_report.txt. The report does not print the summary section. It includes a max-delay analysis and only reports paths with a slack value less than 0.50 ns. It reports a maximum of 3 paths per section and does not report any expanded paths. It only reports timing information for the clock domains count8_clock and count2_clk.

```
report -type timing -print_summary no \
-analysis max \
-use_slack_threshold yes \
-slack_threshold 0.50 \
-print_paths yes -max_paths 3 \
-max_expanded_paths 0 \
-include_user_sets yes \
-include_pin_to_pin yes \
-select_clock_domains yes \
-clock_domain {count8_clock count2_clk} \
timing_report.txt
```

See Also

Tcl documentation conventions

`report (Timing violations) using SmartTime`

`report (Datasheet) using SmartTime`

Designer Tcl Command Reference

report (Timing violations) using SmartTime

Tcl command; creates a timing violations report.

```
report -type timing_violations \
[-analysis value]\
[-use_slack_threshold value]\
[-slack_threshold value]\
[-limit_max_paths value]\
[-max_paths value]\
[-max_expanded_paths value] \
[-format value]
filename
```

Arguments

`-type timing_violations`

Specifies the type of report to generate.

`-analysis value`

Specifies whether to consider minimum analysis or maximum analysis in the timing violations report.

Value	Description
min	Timing report considers minimum analysis
max	Timing report considers maximum analysis (the default value)

`-use_slack_threshold value`

Specifies whether to consider the slack threshold in the timing violations report.

Value	Description
yes	Includes slack threshold in the timing violations report
no	Excludes slack threshold in the timing violations report (the default value)

`-slack_threshold value`

Specifies the threshold to consider when reporting path slacks. This value is a floating-point number in nanoseconds (ns). By default, there is no threshold (all slacks reported).

`-limit_max_paths value`

Specifies if the paths are limited by the number of paths.

Value	Description
yes	Limits the maximum number of paths to report
no	Specifies that there is no limit to the number of paths to report (the default value)

`-max_paths value`

Specifies the maximum number of paths to display for each set. This value is a positive integer value greater than zero. Default is 100.

`-max_expanded_paths value`

Specifies the number of paths to expand per set. This value is a positive integer value greater than zero. The default is 0.

`-format value`

Specifies the output format of the generated report.

Value	Description
text	Generates a text report; text is the default value
csv	Generates the report in a comma-separated value format which you can import into a spreadsheet

`filename`

Specifies the name and destination of the timing violations report.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

The following example generates a timing violations report named timg_viol.txt. The report considers an analysis using maximum delays and does not filter paths based on slack threshold. It reports 2 paths per section and 1 expanded path per section.

```
report -type timing_violations \
-analysis max -use_slack_threshold no \
-limit_max_paths -yes \
-max_paths 2 \
-max_expanded_paths 1 \
timg_viol.txt
```

See Also

Tcl documentation conventions

`report (Timing) using SmartTime`

`report (Datasheet) using SmartTime`

Designer Tcl Command Reference

save

Tcl command; saves all changes made prior to this command. This includes changes made on constraints, options and sets.

```
save
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following script sets the maximum number of paths reported by list_paths to 10, reads an SDC file, and save both the option and the constraints into the design project:

```
set_options -limit_max_paths 10
read_sdc somefile.sdc
save
```

See Also

set_options

save_design

Tcl command; the save_design command saves the current design in Designer to a file. If filename is not a complete path name, the ADB file is written into the current working directory.

```
save_design filename
```

Arguments

The design is written to a file denoted by the variable filename as an ADB file.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Example 1: Saves the design to a file “test.adb” in the current folder.

```
save_design {test.adb}
```

Example 2: Save design and check if it saved successfully.

```
set designFile {d:/test/my_design.adb}
if  { [catch { save_design $designFile }] {
              Puts “Failed to save design”
              # Handle Failure
} else {
              puts “Design saved successfully”
              # Proceed to make further changes
}
```

See Also

close_design

new_design

open_design

Designer Tcl Command Reference

set_clock_latency

Tcl command; defines the delay between an external clock source and the definition pin of a clock within SmartTime.

```
set_clock_latency -source [-rise][-fall][-early][-late] delay clock
```

Arguments

`-source`

Specifies the source latency on a clock pin, potentially only on certain edges of the clock.

`-rise`

Specifies the edge for which this constraint will apply. If neither or both rise are passed, the same latency is applied to both edges.

`-fall`

Specifies the edge for which this constraint will apply. If neither or both rise are passed, the same latency is applied to both edges.

`-invert`

Specifies that the generated clock waveform is inverted with respect to the reference clock.

`-late`

Optional. Specifies that the latency is late bound on the latency. The appropriate bound is used to provide the most pessimistic timing scenario. However, if the value of "-late" is less than the value of "-early", optimistic timing takes place which could result in incorrect analysis. If neither or both "-early" and "-late" are provided, the same latency is used for both bounds, which results in the latency having no effect for single clock domain setup and hold checks.

`-early`

Optional. Specifies that the latency is early bound on the latency. The appropriate bound is used to provide the most pessimistic timing scenario. However, if the value of "-late" is less than the value of "-early", optimistic timing takes place which could result in incorrect analysis. If neither or both "-early" and "-late" are provided, the same latency is used for both bounds, which results in the latency having no effect for single clock domain setup and hold checks.

`delay`

Specifies the latency value for the constraint.

`clock`

Specifies the clock to which the constraint is applied. This clock must be constrained.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Clock source latency defines the delay between an external clock source and the definition pin of a clock within SmartTime. It behaves much like an input delay constraint. You can specify both an "early" delay and a"late" delay for this latency, providing an uncertainty which SmartTime propagates through its calculations. Rising and falling edges of the same clock can have different latencies. If only one value is provided for the clock source latency, it is taken as the exact latency value, for both rising and falling edges.

Examples

The following example sets an early clock source latency of 0.4 on the rising edge of main_clock. It also sets a clock source latency of 1.2, for both the early and late values of the falling edge of main_clock. The late value for the clock source latency for the falling edge of main_clock remains undefined.

```
set_clock_latency –source –rise –early 0.4 { main_clock  }
set_clock_latency –source –fall    1.2 { main_clock }
```

See Also

`create_clock`

`create_generated_clock`

Tcl Command Documentation Conventions

Designer Tcl Command Reference

set_clock_uncertainty

Tcl command; specifies a clock-to-clock uncertainty between two clocks (from and to) and returns the ID of the created constraint if the command succeeded.

```
set_clock_uncertainty uncertainty -from | -rise_from | -fall_from from_clock_list -to | -
rise_to | -fall_to to_clock_list -setup {value} -hold {value}
```

Arguments

uncertainty

Specifies the time in nanoseconds that represents the amount of variation between two clock edges.

-from

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the source clock list. Only one of the -from, -rise_from, or -fall_from arguments can be specified for the constraint to be valid.

-rise_from

Specifies that the clock-to-clock uncertainty applies only to rising edges of the source clock list. Only one of the -from, -rise_from, or -fall_from arguments can be specified for the constraint to be valid.

-fall_from

Specifies that the clock-to-clock uncertainty applies only to falling edges of the source clock list. Only one of the -from, -rise_from, or -fall_from arguments can be specified for the constraint to be valid.

from_clock_list

Specifies the list of clock names as the uncertainty source.

-to

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the destination clock list. Only one of the -to, -rise_to , or -fall_to arguments can be specified for the constraint to be valid.

-rise_to

Specifies that the clock-to-clock uncertainty applies only to rising edges of the destination clock list. Only one of the -to, -rise_to , or -fall_to arguments can be specified for the constraint to be valid.

-fall_to

Specifies that the clock-to-clock uncertainty applies only to falling edges of the destination clock list. Only one of the -to, -rise_to , or -fall_to arguments can be specified for the constraint to be valid.

to_clock_list

Specifies the list of clock names as the uncertainty destination.

-setup

Specifies that the uncertainty applies only to setup checks. If none or both -setup and -hold are present, the uncertainty applies to both setup and hold checks.

-hold

Specifies that the uncertainty applies only to hold checks. If none or both -setup and -hold are present, the uncertainty applies to both setup and hold checks.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

The set_clock_uncertainty command sets the timing uncertainty between two clock waveforms or maximum clock skew. Timing between clocks have no uncertainty unless you specify it.

Examples

```
set_clock_uncertainty 10 -from Clk1 -to Clk2
set_clock_uncertainty 0 -from Clk1 -fall_to { Clk2 Clk3 } -setup
set_clock_uncertainty 4.3 -fall_from { Clk1 Clk2 } -rise_to *
set_clock_uncertainty 0.1 -rise_from [ get_clocks { Clk1 Clk2 } ] -fall_to { Clk3 Clk4 }
-setup
set_clock_uncertainty 5 -rise_from Clk1 -to [ get_clocks {*} ]
```

See Also

create_clock

create_generated_clock

remove_clock_uncertainty

Designer Tcl Command Reference

set_current_scenario

Tcl command; specifies the timing scenario for the Timing Analyzer to use. All commands that follow this command will apply to the specified timing scenario.

```
set_current_scenario name
```

Arguments

name

Specifies the name of the timing scenario to which to apply all commands from this point on.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

A timing scenario is a set of timing constraints used with a design. If the specified scenario is already the current one, this command has no effect.

After setting the current scenario, constraints can be listed, added, or removed, the checker can be invoked on the set of constraints, and so on.

This command uses the specified timing scenario to compute timing analysis.

Example

```
set_current_scenario scenario_A
```

See Also

get_current_scenario

Tcl Command Documentation Conventions

Designer Tcl Command Reference

set_defvar (Designer Only)

Tcl command; the set_defvar command sets an internal variable in the Designer system. You must specify at least one argument for this command.

```
set_defvar variable value
```

Arguments

Variable must be a valid Designer internal variable and could be accompanied by an optional value. If the *value* is provided, the *variable* is set the value. If the *value* is null the *variable* is reset.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Example 1:

```
set_defvar “FORMAT” “VHDL”
Sets the FORMAT internal variable to VHDL.
```

Example 2:

```
set variableToSet "DESIGN"
set valueOfVariable “VHDL”
set_defvar $variableToSet $valueOfVariable
```

These commands set the FORMAT variable to VHDL, shows the use of variables for this command.

See Also

get_defvar

Designer Tcl Command Reference

set_design

Tcl command; this set_design command specifies the design name, family and path in which Designer will process the design. This step is absolutely required before importing the source files.

```
set_design -name design_name -family family_name –pathpath_name
```

Note: You need all three arguments for this command to set up your design.

Arguments

`-name design_name`

The name of the design. This is used as the base name for most of the files generated from Designer.

`-family family_name`

The device family for which the design is being targeted.

`-path path_name`

The physical path of the directory in which the design files will be created.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Example 1: Sets up the design and checks if there are any errors

```
set_design -name "test" -family "Axcelerator" -path {.}
set desName “test
set famName “ACT3”
set path {d:/examples/test}

if  { [catch { set_design -name $desName -family $famName -path $path }] {
              Puts “Failed setup design”
              # Handle Failure
} else {
              puts “Design setup successful”
              # Proceed to Import source files
}
```

See Also

new_design

set_device

Designer Tcl Command Reference

set_device

Tcl command; the set_device command specifies the type of device and its parameters. You must specify at least one option for this command. Some of the options may not apply for certain families that do not support the features.

```
set_device -family family_name -die die_name -package package_name -speed speed_grade -voltage
voltage -voltrange volt_range -temprange temp_range -iostd default_io_std -pci value -jtag value
-probe value -trst value -radexp value  -vcci_1.2_voltrange value -vcci_1.2_widerange value -
vcci_1.5_voltrange value -vcci_1.8_voltrange value -vcci_2.5_voltrange value -
vcci_3.3_voltrange value -vcci_3.3_widerange value
```

Arguments

`-family family_name`

Specifies the name of the FPGA device family.

`-die die_name`

Specifies the part name.

`-package package_name`

Specifies the selected package for the device.

`-speed speed_grade`

`Specifies the speed grade of the part.`

`-voltage voltage`

Specifies the core voltage of the device. You can also use it to define the I/O voltage of the part. For example, if you are using a RTSX with a 3.3 to 2.5 voltage, you can use

-voltage 3.3/2.5

`-voltrange volt_range`

Specifies the voltage range to be applied for the device. It is generally MIL, COM and IND denoting Military, Commercial and Industrial respectively.

Alternatively, you can also specify custom values for Best, Typical, and Worst: `-voltrange "1.60 1.50 1.40"`

`-temprange temp_range`

Specifies the temperature range to be applied for the device. Temperature ranges are MIL, COM and IND denoting Military, Commercial and Industrial respectively. Automotive applications generally use the Automotive, TGrade1, or TGrade2 temperature range.

`-iostd default_io_std`

Specifies the default I/O standard of the part.

`-pci value`

Used if the device needs to configure the I/Os for PCI specification. This parameter is equivalent to setting your I/O attributes to PCI in the Project Settings. Values are summarized in the table below.

Value	Description
yes	Device is configured for PCI specification
no	Device is not configured for PCI specification

`-jtag value`

Specifies if pins need to be reserved for JTAG. Values are summarized in the table below.

Value	Description
yes	Pins are reserved for JTAG

Value	Description
no	Pins are not reserved for JTAG

`-probe` *value*

Specifies if the pins need to be preserved for probing. Values are summarized in the table below.

Value	Description
yes	Pins are preserved for probing
no	Pins not preserved for probing

`-trst` *value*

Specifies if the pins need to be reserved for JTAG test reset. Values are summarized in the table below.

Value	Description
yes	Pins are preserved for JTAG test reset
no	Pins are not preserved for JTAG test reset

`-radexp` *value*

Specifies the radiation value (in Krad) for radiation tolerant devices.

`-vcci_1.2_voltrange` *value* `-vcci_1.5_voltrange`*value* `-vcci_1.8_voltrange`*value* `-vcci_2.5_voltrange`*value*`-vcci_3.3_voltrange`*value*

Specifies the voltage range for VCCIx.x. Values are summarized in the table below.

Value	Description
MIL	Sets the voltage range for VCCIx.x to Military
COM	Sets the voltage range for VCCIx.x to Commercial
IND	Sets the voltage range for VCCIx.x to Industrial

Alternatively, you can also specify custom values for Best, Typical, and Worst: `-vcci_x.x_voltrange "1.26 1.20 1.14"`

`-vcci_1.2_widerange` *value*

Specifies the voltage range for VCCI1.2 as wide range. Values are summarized in the table below.

Value	Description
yes	Specifies the voltage range for VCCI1.2 as wide range and sets the def variable IS_VCCI_1.2_WR as "1"
no	Does not specify the voltage range for VCCI1.2 as wide range

`-vcci_3.3_widerange` *value*

Specifies the voltage range for VCCI3.3 as wide range. Values are summarized in the table below.

Value	Description

Value	Description
yes	Specifies the voltage range for VCCI3.3 as wide range and sets the def variable IS_VCCI_3.3_WR as "1"
no	Does not specify the voltage range for VCCI3.3 as wide range

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Example 1: Setting up a design.

```
set_device -die "APA075" -package "208 PQFP" -speed "STD" -voltage "2.5" \
-jtag "yes" -trst "yes" -temprange "COM" -voltrange "COM"\
-vcci_1.2_voltrange "COM" -vcci_1.2_widerange "no" -vcci_1.5_voltrange "1.60 1.50 1.40"
```

See Also

new_design

set_design

Designer Tcl Command Reference

set_disable_timing

Tcl command; disables timing arcs within a cell and returns the ID of the created constraint if the command succeeded.

```
set_disable_timing -from value -to value name
```

Arguments

`-from` *`from_port`*

Specifies the starting port. The –from and –to arguments must either both be present or both omitted for the constraint to be valid.

`-to` *`to_port`*

Specifies the ending port. The –from and –to arguments must either both be present or both omitted for the constraint to be valid.

`name`

Specifies the cell name where the timing arcs will be disabled.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
set_disable_timing -from A -to Y a2
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

set_false_path

Tcl command; identifies paths that are considered false and excluded from the timing analysis in the current timing scenario.

```
set_false_path [-from from_list] [-through through_list] [-to to_list]
```

Arguments

`-from from_list`

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through through_list`

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

`-to to_list`

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

The `set_false_path` command identifies specific timing paths as being false. The false timing paths are paths that do not propagate logic level changes. This constraint removes timing requirements on these false paths so that they are not considered during the timing analysis. The path starting points are the input ports or register clock pins, and the path ending points are the register data pins or output ports. This constraint disables setup and hold checking for the specified paths.

The false path information always takes precedence over multiple cycle path information and overrides maximum delay constraints. If more than one object is specified within one -through option, the path can pass through any objects.

You must specify at least one of the `–from`, `-to`, or `–through` arguments for this constraint to be valid.

Examples

The following example specifies all paths from clock pins of the registers in clock domain clk1 to data pins of a specific register in clock domain clk2 as false paths:

```
set_false_path –from [get_clocks {clk1}] –to reg_2:D
```

The following example specifies all paths through the pin U0/U1:Y to be false:

```
set_false_path -through U0/U1:Y
```

See Also

Tcl Command Documentation Conventions

Designer Tcl Command Reference

set_input_delay

Tcl command; creates an input delay on a port list by defining the arrival time of an input relative to a clock in the current scenario.

```
set_input_delay delay_value -clock clock_ref [–max] [–min] [–clock_fall] input_list
```

Arguments

delay_value

Specifies the arrival time in nanoseconds that represents the amount of time for which the signal is available at the specified input after a clock edge.

`-clock clock_ref`

Specifies the clock reference to which the specified input delay is related. This is a mandatory argument. If you do not specify -max or -min options, the tool assumes the maximum and minimum input delays to be equal.

`-max`

Specifies that delay_value refers to the longest path arriving at the specified input. If you do not specify -max or -min options, the tool assumes maximum and minimum input delays to be equal.

`-min`

Specifies that delay_value refers to the shortest path arriving at the specified input. If you do not specify -max or -min options, the tool assumes maximum and minimum input delays to be equal.

`-clock_fall`

Specifies that the delay is relative to the falling edge of the clock reference. The default is the rising edge.

`input_list`

Provides a list of input ports in the current design to which delay_value is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.and IGLOOe, except ProASIC3 nano and ProASIC3L

Description

The set_input_delay command sets input path delays on input ports relative to a clock edge. This usually represents a combinational path delay from the clock pin of a register external to the current design. For in/out (bidirectional) ports, you can specify the path delays for both input and output modes. The tool adds input delay to path delay for paths starting at primary inputs.

A clock is a singleton that represents the name of a defined clock constraint. This can be:

- a single port name used as source for a clock constraint
- a single pin name used as source for a clock constraint; for instance reg1:CLK. This name can be hierarchical (for instance toplevel/block1/reg2:CLK)
- an object accessor that will refer to one clock: [get_clocks {clk}]

Examples

The following example sets an input delay of 1.2ns for port data1 relative to the rising edge of CLK1:

```
set_input_delay 1.2 -clock [get_clocks CLK1] [get_ports data1]
```

The following example sets a different maximum and minimum input delay for port IN1 relative to the falling edge of CLK2:

```
set_input_delay 1.0 -clock_fall -clock CLK2 –min {IN1}
set_input_delay 1.4 -clock_fall -clock CLK2 –max {IN1}
```

See Also

`set_output_delay`

Tcl Command Documentation Conventions

Designer Tcl Command Reference

set_max_delay

Tcl command; specifies the maximum delay for the timing paths in the current scenario.

```
set_max_delay delay_value [-from from_list] [-to to_list] [-through through_list]
```

Arguments

delay_value

Specifies a floating point number in nanoseconds that represents the required maximum delay value for specified paths.

- If the path starting point is on a sequential device, the tool includes clock skew in the computed delay.
- If the path starting point has an input delay specified, the tool adds that delay value to the path delay.
- If the path ending point is on a sequential device, the tool includes clock skew and library setup time in the computed delay.
- If the ending point has an output delay specified, the tool adds that delay to the path delay.

`-from` *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-to` *to_list*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-through` *through_list*

Specifies a list of pins, ports, cells, or nets through which the timing paths must pass.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

This command specifies the required maximum delay for timing paths in the current design. The path length for any startpoint in from_list to any endpoint in to_list must be less than delay_value.

The timing engine automatically derives the individual maximum delay targets from clock waveforms and port input or output delays.

The maximum delay constraint is a timing exception. This constraint overrides the default single cycle timing relationship for one or more timing paths. This constraint also overrides a multicycle path constraint.

You must specify at least one of the `–from`, `-to`, or `–through` arguments for this constraint to be valid.

Examples

The following example sets a maximum delay by constraining all paths from ff1a:CLK or ff1b:CLK to ff2e:D with a delay less than 5 ns:

```
set_max_delay 5 -from {ff1a:CLK ff1b:CLK} -to {ff2e:D}
```

The following example sets a maximum delay by constraining all paths to output ports whose names start by “out” with a delay less than 3.8 ns:

```
set_max_delay 3.8 -to [get_ports out*]
```

See Also

`set_min_delay`

`remove_max_delay`

Tcl Command Documentation Conventions

Designer Tcl Command Reference

set_min_delay

Tcl command; specifies the minimum delay for the timing paths in the current scenario.

```
set_min_delay delay_value [-from from_list] [-to to_list] [-through through_list]
```

Arguments

delay_value

Specifies a floating point number in nanoseconds that represents the required minimum delay value for specified paths.

- If the path starting point is on a sequential device, the tool includes clock skew in the computed delay.
- If the path starting point has an input delay specified, the tool adds that delay value to the path delay.
- If the path ending point is on a sequential device, the tool includes clock skew and library setup time in the computed delay.
- If the ending point has an output delay specified, the tool adds that delay to the path delay.

`-from` *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-to` *to_list*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-through` *through_list*

Specifies a list of pins, ports, cells, or nets through which the timing paths must pass.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

This command specifies the required minimum delay for timing paths in the current design. The path length for any startpoint in from_list to any endpoint in to_list must be less than delay_value.

The timing engine automatically derives the individual minimum delay targets from clock waveforms and port input or output delays.

The minimum delay constraint is a timing exception. This constraint overrides the default single cycle timing relationship for one or more timing paths. This constraint also overrides a multicycle path constraint.

You must specify at least one of the `–from`, `-to`, or `–through` arguments for this constraint to be valid.

Examples

The following example sets a minimum delay by constraining all paths from ff1a:CLK or ff1b:CLK to ff2e:D with a delay less than 5 ns:

```
set_min_delay 5 -from {ff1a:CLK ff1b:CLK} -to {ff2e:D}
```

The following example sets a minimum delay by constraining all paths to output ports whose names start by “out” with a delay less than 3.8 ns:

```
set_min_delay 3.8 -to [get_ports out*]
```

See Also

set_max_delay

remove_min_delay

Tcl Command Documentation Conventions

Designer Tcl Command Reference

set_multicycle_path

Tcl command; defines a path that takes multiple clock cycles in the current scenario.

```
set_multicycle_path ncycles [-setup] [-hold] [-from from_list[–through through_list[-to
to_list
```

Arguments

ncycles

Specifies an integer value that represents a number of cycles the data path must have for setup or hold check. The value is relative to the starting point or ending point clock, before data is required at the ending point.

`-setup`

Optional. Applies the cycle value for the setup check only. This option does not affect the hold check. The default hold check will be applied unless you have specified another set_multicycle_path command for the hold value.

`-hold`

Optional. Applies the cycle value for the hold check only. This option does not affect the setup check.

Note: If you do not specify "-setup" or "-hold", the cycle value is applied to the setup check and the default hold check is performed (*ncycles* -1).

`-from` *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through` *through_list*

Specifies a list of pins or ports through which the multiple cycle paths must pass.

`-to` *to_list*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Setting multiple cycle paths constraint overrides the single cycle timing relationships between sequential elements by specifying the number of cycles that the data path must have for setup or hold checks. If you change the multiplier, it affects both the setup and hold checks.

False path information always takes precedence over multiple cycle path information. A specific maximum delay constraint overrides a general multiple cycle path constraint.

If you specify more than one object within one -through option, the path passes through any of the objects.

You must specify at least one of the `–from`, `-to`, or `–through` arguments for this constraint to be valid.

Exceptions

Multiple priority management is not supported in Microsemi SoC designs. All multiple cycle path constraints are handled with the same priority.

Examples

The following example sets all paths between reg1 and reg2 to 3 cycles for setup check. Hold check is measured at the previous edge of the clock at reg2.

```
set_multicycle_path 3 -from [get_pins {reg1}] –to [get_pins {reg2}]
```

The following example specifies that four cycles are needed for setup check on all paths starting at the registers in the clock domain ck1. Hold check is further specified with two cycles instead of the three cycles that would have been applied otherwise.

```
set_multicycle_path 4 -setup -from [get_clocks {ck1}]
set_multicycle_path 2 -hold -from [get_clocks {ck1}]
```

See Also

remove_multicycle_path

Tcl Command Documentation Conventions

Designer Tcl Command Reference

set_output_delay

Tcl command; defines the output delay of an output relative to a clock in the current scenario.

```
set_output_delay delay_value -clock clock_ref [–max] [–min] [–clock_fall] output_list
```

Arguments

delay_value

Specifies the amount of time before a clock edge for which the signal is required. This represents a combinational path delay to a register outside the current design plus the library setup time (for maximum output delay) or hold time (for minimum output delay).

-clock *clock_ref*

Specifies the clock reference to which the specified output delay is related. This is a mandatory argument. If you do not specify -max or -min options, the tool assumes the maximum and minimum input delays to be equal.

-max

Specifies that delay_value refers to the longest path from the specified output. If you do not specify -max or -min options, the tool assumes the maximum and minimum output delays to be equal.

-min

Specifies that delay_value refers to the shortest path from the specified output. If you do not specify -max or -min options, the tool assumes the maximum and minimum output delays to be equal.

-clock_fall

Specifies that the delay is relative to the falling edge of the clock reference. The default is the rising edge.

output_list

Provides a list of output ports in the current design to which delay_value is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

The set_output_delay command sets output path delays on output ports relative to a clock edge. Output ports have no output delay unless you specify it. For in/out (bidirectional) ports, you can specify the path delays for both input and output modes. The tool adds output delay to path delay for paths ending at primary outputs.

Examples

The following example sets an output delay of 1.2ns for port OUT1 relative to the rising edge of CLK1:

```
set_output_delay 1.2 -clock [get_clocks CLK1] [get_ports OUT1]
```

The following example sets a different maximum and minimum output delay for port OUT1 relative to the falling edge of CLK2:

```
set_output_delay 1.0 -clock_fall -clock CLK2 –min {OUT1}
set_output_delay 1.4 -clock_fall -clock CLK2 –max {OUT1}
```

See Also

remove_output_delay

set_input_delay

Tcl Command Documentation Conventions

Designer Tcl Command Reference

smartpower_add_new_custom_mode

Tcl command; creates a new custom mode.

```
smartpower_add_new_custom_mode -name {mode_name} -base_mode {base_mode} -description
{mode_description}
```

Arguments

`-name {mode_name}`

Specifies the name of the new custom mode.

`-base_mode {base_mode}`

Specifies the name of the base mode used to create the new custom mode. It must be one of the following: Active, Standby, or Flash*Freeze.

`-description {mode_description}`

Specifies the description of the new custom mode.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example creates a new custom mode "Cust_1" based on the Active mode:

```
smartpower_add_new_custom_mode -name {Cust_1} -base_mode {Active} -description
{frequency 10 MHz}
```

See Also

smartpower_remove_custom_mode

Designer Tcl Command Reference

smartpower_add_new_scenario

Tcl command; creates a new scenario.

```
smartpower_add_new_scenario -name {value} -description {value} -mode {value}
```

Arguments

`-name {value}`

Specifies the name of the new scenario.

`-description {value}`

Specifies the description of the new scenario.

`-mode {<operating mode>:<duration>}+`

Specifies the mode(s) and duration(s) for the specified scenario.

Supported Families

See the Tcl Commands and Supported Families table for a list of supported families.

Examples

This example creates a new scenario called myscenario:

```
smartpower_add_new_scenario -name "MyScenario" -mode "Custom_1:50.00"
"Custom_2:25.00" -mode "Active:25.00"
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_add_pin_in_domain

Tcl command; adds a pin into a clock or set domain.

```
smartpower_add_pin_in_domain -pin_name {pin_name} -pin_type {value} –domain_name
{domain_name} -domain_type {value}
```

Arguments

`-pin_name {pin_name}`

Specifies the name of the pin to add to the domain.

`-pin_type {value}`

Specifies the type of the pin to add. The following table shows the acceptable values for this argument:

Value	Description
clock	The pin to add is a clock pin
data	The pin to add is a data pin

`-domain_name {domain_name}`

Specifies the name of the domain in which to add the specified pin.

`-domain_type {value}`

Specifies the type of domain in which to add the specified pin. The following table shows the acceptable values for this argument:

Value	Description
clock	The domain is a clock domain

Value	Description
set	The domain is a set domain

Supported Families

See the Tcl Commands and Supported Families table for a list of supported families.

Notes

- The `domain_name` must be a name of an existing domain.
- The `pin_name` must be a name of a pin that exists in the design.

Examples

The following example adds a clock pin to an existing Clock domain:

```
smartpower_add_pin_in_domain -pin_name { XCMP3/U0/U1:Y } -pin_type {clock} –domain_name {clk1} -domain_type {clock}
```

The following example adds a data pin to an existing Set domain:

```
smartpower_add_pin_in_domain -pin_name {XCMP3/U0/U1:Y} -pin_type {data} -domain_name {myset} -domain_type {set}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

`smartpower_remove_pin_of_domain`

smartpower_change_clock_statistics

Tcl command; changes the default frequencies and probabilities for a specific domain.

```
smartpower_change_clock_statistics -domain_name {value} -clocks_freq {value} -clocks_proba {value} -registers_freq {value} -registers_proba {value} -set_reset_freq {value} -set_reset_proba {value} -primaryinputs_freq {value} -primaryinputs_proba {value} -combinational_freq {value} -combinational_proba {value}
```

Arguments

`-domain_name{value}`

Specifies the domain name in which to initialize frequencies and probabilities.

`-clocks_freq {value}`

Specifies the user input frequency in Hz, KHz, or MHz for all clocks.

`-clocks_proba {value}`

Specifies the user input probability in % for all clocks.

`-registers_freq {value}`

Specifies the user input frequency (in Hz, KHz, or MHz) or the toggle rate (in %). If the unit is not provided and toggle rate is active, the value is handled as a toggle rate; if toggle rate is not active, the value is handled as a frequency.

`-registers_proba {value}`

Specifies the user input probability in % for all registers.

`-set_reset_freq {value}`

Specifies the user input frequency (in Hz, KHz, or MHz) or the toggle rate (in %). If the unit is not provided and toggle rate is active, the value is handled as a toggle rate; if toggle rate is not active, the value is handled as a frequency.

`-set_reset_proba {value}`

Specifies the user input probability in % for all set/reset nets.

`-primaryinputs_freq {value}`

Specifies the user input frequency (in Hz, KHz, or MHz) or the toggle rate (in %). If the unit is not provided and toggle rate is active, the value is handled as a toggle rate; if toggle rate is not active, the value is handled as a frequency.

`-primaryinputs_proba {value}`

Specifies the user input probability in % for all primary inputs.

`-combinational_freq {value}`

Specifies the user input frequency (in Hz, KHz, or MHz) or the toggle rate (in %). If the unit is not provided and toggle rate is active, the value is handled as a toggle rate; if toggle rate is not active, the value is handled as a frequency.

`-combinational_proba {value}`

Specifies the user input probability in % for all combinational combinational output.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all clocks withs:

```
smartpower_change_clock_statistics -domain_name {my_domain} -clocks_freq {10 MHz} -
clocks_proba {20} -registers_freq {10 MHz} -registers_proba {20} -set_reset_freq {10
MHz} -set_reset_proba {20} -primaryinputs_freq {10 MHz} -primaryinputs_proba {20} -
combinational_freq {10 MHz} -combinational_proba {20}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_change_setofpin_statistics

Tcl command; changes the default frequencies and probabilities for a specific set.

```
smartpower_change_setofpin_statistics -domain_name {value} -data_freq {value} -
data_proba {value}
```

Arguments

`-domain_name{value}`

Specifies the domain name in which to initialize data frequencies and probabilities.

`-data_freq {value}`

Specifies the user input data frequency in Hz, KHz, or MHz for all sets of pins.

`-data_proba {value}`

Specifies the user input data probability in % for all sets of pins.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all clocks withs:

```
smartpower_change_setofpin_statistics -domain_name {my_domain} -data_freq {10 MHz} -
data_proba {20}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_commit

Tcl command; saves the changes to the design (.adb) file.

```
smartpower_commit
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
smartpower_commit
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_restore

smartpower_compute_vectorless

This Tcl command executes a vectorless analysis of the current operating mode.

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
smartpower_compute_vectorless
```

See Also

Tcl Command Documentation Conventions
Designer Tcl Command Reference

smartpower_create_domain

Tcl command; creates a new clock or set domain.

```
smartpower_create_domain -domain_type {value} -domain_name {domain_name}
```

Arguments

`-domain_type {value}`

Specifies the type of domain to create. The following table shows the acceptable values for this argument:

Value	Description
clock	The domain is a clock domain
set	The domain is a set domain

`-domain_name {domain_name}`

Specifies the name of the new domain.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

The `domain name` cannot be the name of an existing domain.

The domain `type` must be either clock or set.

Examples

The following example creates a new clock domain named "clk2":

```
smartpower_create_domain -domain_type {clock} -domain_name {clk2}
```

The following example creates a new set domain named "myset":

```
smartpower_create_domain -domain_type {set} -domain_name {myset}
```

See Also

Tcl documentation conventions
Designer Tcl Command Reference
`smartpower_remove_domain`

smartpower_edit_custom_mode

Tcl command; edits a custom mode.

```
smartpower_edit_custom_mode -name {old_mode_name} new_name {new_mode_name} -description
{mode_description}
```

Arguments

`-name {old_mode_name}`

Specifies the name of the custom mode you want to edit.

`-new_name {new_mode_name}`

Specifies the new name of the custom mode.

`-description {mode_description}`

Specifies the description of the new custom mode.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example edits custom mode "Cust_1" and renames it "Cust_2":

```
smartpower_edit_custom_mode -name {Cust_1} -new_name {Cust_2} -description {frequency 10 MHz}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

`smartpower_remove_custom_mode`

`smartpower_add_custom_mode`

smartpower_edit_scenario

Tcl command; edits a scenario.

```
smartpower_edit_scenario -name {value} -description {value} -mode {value} -new_name {value}
```

Arguments

`-name {value}`

Specifies the name of the scenario.

`-description {value}`

Specifies the description of the scenario.

`-mode {<operating mode>:<duration>}`

Specifies the mode(s) and duration(s) for the specified scenario.

`-new_name {value}`

Specifies the new name for the scenario

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example edits the name of myscenario to finalscenario:

```
smartpower_edit_scenario -name myscenario -new_name finalscenario
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_import_vcd

This SmartPower Tcl command imports into SmartPower a VCD file generated by a simulation tool. SmartPower extracts the frequency and probability information from the VCD.

```
import_vcd -file "VCD file" [-opmode "mode name"] [-with_vectorless "TRUE | FALSE"] [-
partial_parse\ "TRUE | FALSE"] [-start_time "decimal value"] [-end_time "decimal value"]
\
[-auto_detect_top_level_name "TRUE | FALSE"] [-top_level_name "top level name"] [-
glitch_filtering\ "false | auto | true"] [-glitch_threshold "integer value"] [-stop_time
"decimal value"]
```

Parameters

`-file "VCD file"`

Value must be a file path. This parameter is mandatory.

`[-opmode "mode name"]`

Value must be a string. This parameter is optional.

`[-with_vectorless "TRUE | FALSE"]`

Value must be a boolean. This parameter is optional.

`[-partial_parse "TRUE | FALSE"]`

Value must be a boolean. This parameter is optional.

`[-start_time "decimal value"]`

Value must be a positive decimal. This parameter is optional.

`[-end_time "decimal value"]`

Value must be a positive decimal. This parameter is optional.

`[-auto_detect_top_level_name "TRUE | FALSE"]`

Value must be a boolean. This parameter is optional.

`[-top_level_name "top level name"]`

Value must be a string. This parameter is optional.

`[-glitch_filtering "false | auto | true"]`

Value must be one of false | auto | true. This parameter is optional.

`[-glitch_threshold "integer value"]`

Value must be a positive integer. This parameter is optional.

Exceptions

None

Returns

This command does not return a value.

Usage

This section lists all the parameters for the command, their types, and the values they can be set to. The default value is always listed first.

smartpower_import_vcd	Type	Values	Description
file	String	Path to a VCD file	Path to a VCD file.
opmode	String	Operating mode name “Active” by default	Operating mode in which the VCD will be imported. If the mode doesn’t exist, it will be created.
with_vectorless	Boolean	TRUE|FALSE	Specify the method to set the frequency and probability information for signals not annotated by the VCD TRUE:

smartpower_import_vcd	Type	Values	Description
			use the vectorless analysis FALSE: use average value computed from the VCD.
partial_parse	Boolean	FALSE|TRUE	Enable partial parsing of the VCD. Start time and end time need to be specified when TRUE.
start_time	Decimal value	positive decimal nanoseconds (ns)	Specify the starting timestamp of the VCD extraction in ns. It must be lower than the specified end_time. It must be lower than the last timestamp in the VCD file.
end_time	Decimal value	positive decimal nanoseconds (ns)	Specify the end timestamp of the VCD extraction in ns. It must be higher than the specified start_time.
auto_detect_top_level_name	Boolean	TRUE|FALSE	Enable the auto detection of the top level name in the VCD file. Top_level_name needs to be specified when FALSE .
top_level_name	Boolean	Full hierarchical name	Specify the full hierarchical name of the instance of the design in the VCD file.
glitch_filtering	Boolean	Auto|FALSE|TRUE	AUTO: Enable glitch filtering with predefined thereshold based on the family TRUE: Enable glitch filtering, glitch_threshold must be specified FALSE: Disable glitch filtering.
glitch_threshold	Integer	Positive integer	Specify the threshold in ps below which glitches are filtered out.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

The Tcl command below imports the power.vcd file generated by the simulator into SmartPower:

```
smartpower_import_vcd –file “../../simulation/power.vcd”
```

The Tcl command below extracts information between 1ms and 2ms in the simulation, and stores the information into a custom mode:

```
smartpower_import_vcd –file “../../simulation/power.vcd” –partial_parse TRUE –start_time
1000000 –end_time 2000000 –opmode “power_1ms_to_2ms”
```

smartpower_init_do

Tcl command; initializes the frequencies and probabilities for clocks, registers, set/reset nets, primary inputs, combinational outputs, enables and other sets of pins, and selects a mode for initialization.

```
smartpower_init_do -with {value} -opmode {value} -clocks {value} -registers {value} -
set_reset {value} -primaryinputs {value} -combinational {value} -enables {value} -othersets
{value}
```

Arguments

`-with{value}`

This sets the option of initializing frequencies and probabilities with vectorless analysis or with fixed values. The following table shows the acceptable values for this argument:

Value	Description
vectorless	Initializes frequencies and probabilities with vectorless analysis
fixed	Initializes frequencies and probabilities with fixed values

`-opmode {value}`

Optional; specifies the mode in which to initialize frequencies and probabilities. The value must be Active or Flash*Freeze.

`-clocks {value}`

This sets the option of initializing frequencies and probabilities for all clocks. The following table shows the acceptable values for this argument:

Value	Description
true	Initializes frequencies and probabilities for all clocks
false	Does not initialize frequencies and probabilities for all clocks

`-registers {value}`

This sets the option of initializing frequencies and probabilities for all registers. The following table shows the acceptable values for this argument:

Value	Description
true	Initializes frequencies and probabilities for all registers
false	Does not initialize frequencies and probabilities for all registers

`-set_reset {value}`

This sets the option of initializing frequencies and probabilities for all set/reset nets. The following table shows the acceptable values for this argument:

Value	Description
true	Initializes frequencies and probabilities for all set/reset nets
false	Does not initialize frequencies and probabilities for all set/reset nets

`-primaryinputs{value}`

This sets the option of initializing frequencies and probabilities for all primary inputs. The following table shows the acceptable values for this argument:

Value	Description
true	Initializes frequencies and probabilities for all primary inputs
false	Does not initialize frequencies and probabilities for all primary inputs

`-combinational {value}`

This sets the option of initializing frequencies and probabilities for all combinational outputs. The following table shows the acceptable values for this argument:

Value	Description
true	Initializes frequencies and probabilities for all combinational outputs
false	Does not initialize frequencies and probabilities for all combinational outputs

`-enables {value}`

This sets the option of initializing frequencies and probabilities for all enable sets of pins. The following table shows the acceptable values for this argument:

Value	Description
true	Initializes frequencies and probabilities for all enable sets of pins
false	Does not initialize frequencies and probabilities for all enable sets of pins

`-othersets {value}`

This sets the option of initializing frequencies and probabilities for all other sets of pins. The following table shows the acceptable values for this argument:

Value	Description
true	Initializes frequencies and probabilities for all other sets of pins
false	Does not initialize frequencies and probabilities for all other sets of pins

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all clocks with:

```
smartpower_init_do -with {vectorless} -opmode {my_mode} -clocks {true} -registers {true}
-asynchronous {true} -primaryinputs {true} -combinational {true} -enables {true} -
othersets {true}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_init_set_clocks_options

Tcl command; initializes the clock frequency options of all clock domains.

```
smartpower_init_set_clocks_options -with_clock_constraints {value} -
with_default_values {value} -freq {value} -duty_cycle {value}
```

Arguments

`-with_clock_constraints {value}`

This sets the option of initializing the clock frequencies with frequency constraints from SmartTime. The following table shows the acceptable values for this argument:

Value	Description
true	Sets initialize clock frequencies with clock constraints ON
false	Sets initialize clock frequencies with clock constraints OFF

`-with_default_values {value}`

This sets the option of initializing the clock frequencies with a user input default value. The following table shows the acceptable values for this argument:

Value	Description
true	Sets initialize clock frequencies with default values ON
false	Sets initialize clock frequencies with default values OFF

`-freq {value}`

Specifies the user input frequency in Hz, KHz, or MHz.

`-duty_cycle {value}`

Specifies the user input duty cycles in %.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all clocks after executing `smartpower_init_do` with `-clocks {true}`:

```
smartpower_init_set_clocks_options -with_clock_constraints {true} -with_default_values {true} -freq {10 MHz} -duty_cycle {20}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_init_set_combinational_options

Tcl commands; initializes the frequency and probability of all combinational outputs.

```
smartpower_init_set_combinational_options -freq {value} -proba {value}
```

Arguments

`-freq {value}`

Specifies the user input frequency (in Hz, KHz, or MHz) or the toggle rate (in %). If the unit is not provided and toggle rate is active, the value is handled as a toggle rate; if toggle rate is not active, the value is handled as a frequency.

`-proba {value}`

Specifies the user input probability in %.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all combinational signals after executing `smartpower_init_do` with `-combinational {true}`:

```
smartpower_init_set_combinational_options -freq {10 MHz} -proba {20}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_init_set_enables_options

Tcl command; initializes the clock frequency of all enable clocks with the initialization options.

```
smartpower_init_set_enables_options -freq {value} -proba {value}
```

Arguments

`-freq {value}`

Specifies the user input frequency (in Hz, KHz, or MHz).

`-proba {value}`

Specifies the user input probability in %.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all clocks after executing `smartpower_init_do` with `-enables {true}`:

```
smartpower_init_set_enables_options -freq {10 MHz} -proba {20}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_init_set_othersets_options

Tcl command; initializes the frequency and probability of all other sets.

```
smartpower_init_set_othersets_options [-freq "decimal value [ unit { Hz | KHz | MHz } ]"]
[-proba "decimal value"]
[-with "vectorless | default"]
[-input_freq "decimal value [ unit { Hz | KHz | MHz } ]"]
[-input_proba "decimal value"]
```

Arguments

`-freq "decimal value [unit {Hz | KHz| MHz}"`

Specifies the default frequency and units.

`-proba {decimal value}`

Specifies the default probability.

`-with "vectorless | default"`

Specifies vectorless or default analysis.

`-input_freq "decimal value [unit {Hz | KHz| MHz}"`

Specifies the input frequency and units.

`-input_proba {decimal value}`

Specifies the input probability.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

This command is associated with the functionality of Initialize Frequencies and Probabilities dialog box.

Examples

The following example initializes all other sets after executing `smartpower_init_do` with `-othersets {true}`:

```
smartpower_init_set_othersets_options -freq {10 MHz} -proba {20} [-with default]
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_init_set_primaryinputs_options

Tcl command; initializes the frequency and probability of all primary inputs.

```
smartpower_init_set_primaryinputs_options -freq {value} -proba {value}
```

Arguments

`-freq {value}`

Specifies the user input frequency (in Hz, KHz, or MHz) or the toggle rate (in %). If the unit is not provided and toggle rate is active, the value is handled as a toggle rate; if toggle rate is not active, the value is handled as a frequency.

`-proba {value}`

Specifies the user input probability in %.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all primary inputs after executing `smartpower_init_do` with `-primaryinputs {true}`:

```
smartpower_init_set_primaryinputs_options -freq {10 MHz} -proba {20}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_init_set_registers_options

Tcl command; initializes the frequency and probability of all register outputs.

```
smartpower_init_set_registers_options -freq {value} -proba {value}
```

Arguments

`-freq {value}`

Specifies the user input frequency (in Hz, KHz, or MHz) or the toggle rate (in %). If the unit is not provided and toggle rate is active, the value is handled as a toggle rate; if toggle rate is not active, the value is handled as a frequency.

`-proba {value}`

Specifies the user input probability in %.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Exceptions

None

Examples

The following example initializes all register outputs after executing `smartpower_init_do` with `-registers {true}`:

```
smartpower_init_set_registers_options -freq {10 MHz} -proba {20}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_init_set_set_reset_options

Tcl command; initializes the frequency and probability of all set and reset nets.

```
smartpower_init_set_set_reset_options -freq {value} -proba {value}
```

Arguments

`-freq {value}`

Specifies the user input frequency (in Hz, KHz, or MHz) or the toggle rate (in %). If the unit is not provided and toggle rate is active, the value is handled as a toggle rate; if toggle rate is not active, the value is handled as a frequency.

`-proba {value}`

Specifies the user input probability in %.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all set/reset nets after executing `smartpower_init_do` with `-set_reset {true}`:

```
smartpower_init_set_set_reset_options -freq {10 MHz} -proba {20}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_init_setofpins_values

Tcl command; initializes the frequency and probability of all sets of pins.

```
smartpower_init_setofpins_values -domain_name {name} -freq {value} -proba {value}
```

Arguments

`-domain_name{name}`

Specifies the set of pins that will be initialized. The following table shows the acceptable values for this argument:

Value	Description
IOsEnableSet	Specifies that the IOsEnableSet set of pins will be initialized
MemoriesEnableSet	Specifies that the MemoriesEnableSet set of pins will be initialized

`-freq {value}`

Specifies the user input frequency in Hz, MHz, or KHz.

`-proba {value}`

Specifies the user input probability in %.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all primary inputs after executing `smartpower_init_do` with `-othersets {true}`:

```
smartpower_init_setofpins_values -domain_name {IOsEnableSet} -freq {10 MHz} -proba {20}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_remove_all_annotations

Tcl command; removes all initialization annotations for the specified mode.

```
smartpower_remove_all_annotations -opmode {value}
```

Arguments

`-opmode {value}`

Removes all initialization annotations for the specified mode, where value must be Active or Flash*Freeze.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

This command is associated with the functionality of Initialize frequencies and probabilities dialog box.

Examples

The following example initializes all clocks with opmode Acitve:

```
smartpower_remove_all_annotations -opmode {Active}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_remove_custom_mode

Tcl command; removes a custom mode.

```
smartpower_remove_custom_mode -name {deleted_mode_name}
```

Arguments

`-name {deleted_mode_name}`

Specifies the name of the custom mode you want to delete.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example delets custom mode "Cust_1":

```
smartpower_delete_custom_mode -name {Cust_1}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

`sp_add_custom_mode`

`sp_edit_custom_mode`

smartpower_remove_domain

Tcl command; removes an existing clock or set domain.

```
smartpower_remove_domain -domain_type {value} -domain_name {domain_name}
```

Arguments

`-domain_type {value}`

This specifies the type of domain to remove.The following table shows the acceptable values for this argument:

Value	Description

Value	Description
clock	The domain is a clock domain
set	The domain is a set domain

`-domain_name {domain_name}`

This specifies the name of the domain to remove

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

The domain name must be the name of an existing domain.

The domain type must be either clock or set.

Examples

The following example removes the clock domain named "clk2":

```
smartpower_remove_domain -domain_type {clock} -domain_name {clk2}
```

The following example removes the set domain named "myset":

```
smartpower_remove_domain -domain_type {set} -domain_name {myset}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

`smartpower_create_domain`

smartpower_remove_file

Tcl command; removes a VCD file from the specified mode or all operating mode. Frequency and probability information of signals annotated by the VCD are set back to the default value..

```
remove_file
-file {value} \
-format {value} \
-opmode {value} \
```

Arguments

`-file {value}`

Specifies the file to be removed. This is mandatory.

`-format VCD`

Specifies that the type to be removed is a VCD file. This is mandatory.

`[-opmode {value}]`

Specifies the operating mode. This is optional. The following table shows the acceptable values for this argument:

Value	Description
Active	The operating mode is set to active

Value	Description
Standby	The operating mode is set to static
Flash*Freeze	The operating mode is set to Flash*Freeze

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example removes the file test.vcd from the Active mode.

```
smartpower_remove_file -file "test.vcd" -format VCD -opmode "Active"
```

This example removes the VCD file power1.vcd from all operating modes:

```
smartpower_remove_file –file “power1.vcd” –format VCD
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

This command was obsoleted in SmartPower v8.5. Update your script to use `smartpower_remove_pin_probability` to remove the pin probability.

Note: The information below is obsolete and should only be used as reference when executing previously-created scripts. Update your scripts to use `smartpower_remove_pin_probability`.

Removes the probability value associated with a specific pin. This pin will have a default probability based on the domain set it belongs to.

```
smartpower_remove_pin_enable_rate –pin_name {pin_name}
```

Arguments

`-pin_name {pin_name}`

Specifies the name of the pin with the probability to remove. This pin must be the direct driver of an enable pin.

Supported Families

SmartFusion, IGLOO, ProASIC3, Fusion

Exceptions

None

Examples

The following example removes the probability of the pin driving the enable pin of a bidirectional I/O:

```
Smartpower_remove_pin_enable_rate –pin_name mybibuf/U0/U1:EOUT
```

smartpower_remove_pin_frequency

Tcl command; removes the frequency associated with a specific pin. This pin will have a default frequency based on its domain.

```
smartpower_remove_pin_frequency -pin_name {pin_name}
```

Arguments

`-pin_name {pin_name}`

Specifies the name of the pin for which the frequency will be removed.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

The *pin_name* must be the name of a pin that already exists in the design and already belongs to a domain.

Examples

The following example removes the frequency from the pin named "count8_clock":

```
smartpower_remove_pin_frequency -pin_name {count8_clock}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

`smartpower_set_pin_frequency`

smartpower_remove_pin_of_domain

Tcl command; removes a clock pin or a data pin from a clock or set domain, respectively.

```
smartpower_remove_pin_of_domain -pin_name {pin_name} -pin_type {value} -domain_name
{domain_name} -domain_type {value}
```

Arguments

`-pin_name {pin_name}`

Specifies the name of the pin to remove from the domain.

`-pin_type {value}`

Specifies the type of the pin to remove. The following table shows the acceptable values for this argument:

Value	Description
clock	The pin to remove is a clock pin
data	The pinto remove is a data pin

`-domain_name {domain_name}`

Specifies the name of the domain from which to remove the pin.

`-domain_type {value}`

Specifies the type of domain from which the pin is being removed. The following table shows the acceptable values for this argument:

Value	Description
clock	The domain is a clock domain
set	The domain is a set domain

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

The domain name must be the name of an existing domain.

The pin name must be the name of an existing pin.

Examples

The following example removes the clock pin named "XCMP3/UO/U1:Y" from the clock domain named "clockh":

```
smartpower_remove_pin_of_domain -pin_name {XCMP3/U0/U1:Y}
-pin_type {clock} -domain_name {clockh} -domain_type {clock}
```

The following example removes the data pin named "count2_en" from the set domain named "InputSet":

```
smartpower_remove_pin_of_domain -pin_name {count2_en} -pin_type
{data} -domain_name {InputSet} -domain_type {set}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

`smartpower_add_pin_in_domain`

smartpower_remove_pin_probability

Tcl command; removes the probability value associated with a specific pin. This pin will have a default probability based on the domain set it belongs to.

```
smartpower_remove_pin_probability –pin_name {pin_name}
```

Arguments

`-pin_name {pin_name}`

Specifies the name of the pin with the probability to remove. This pin must be the direct driver of an enable pin.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

The following example removes the probability of the pin driving the enable pin of a bidirectional I/O:

```
Smartpower_remove_pin_probability –pin_name mybibuf/U0/U1:EOUT
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

`smartpower_set_pin_probability`

smartpower_remove_scenario

Tcl command; removes a scenario from the current design.

```
smartpower_remove_scenario -name {value}
```

Arguments

`-name {value}`

Specifies the name of the scenario.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example removes a scenario from the current design:

```
smartpower_remove_scenario -name myscenario
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_restore

Tcl command; restores all power information previously committed in SmartPower.

```
smartpower_restore
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
smartpower_restore
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

`smartpower_commit`

smartpower_set_cooling

Tcl command; sets the cooling style to one of the predefined types, or a custom value.

```
smartpower_set_cooling -style {value} -teta {value}
```

Arguments

`-style {value}`

Specifies the cooling style to custom value or to one of the predefined types with a default thermal resistance value. The following table shows the acceptable values for this argument:

Value	Description
300_lfm	Predefined cooling style
case_cooling	Predefined cooling style

Value	Description
still_air	Predefined cooling style
custom	Cooling style defined by user input

`-teta {value}`

Specifies the thermal resistance in °C/W. This argument is only available when style is set to Custom.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

To compute the junction temperature, set the following three commands: smartpower_set_thermalmode, smartpower_set_tambient and smartpower_set_cooling. The junction temperature will be updated when an output command is executed, such as report(Power).

Examples

The following example sets the cooling style to still air:

```
smartpower_set_cooling -style {still_air}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_set_mode_for_analysis

Tcl command; sets the mode for cycle-accurate power analysis.

```
smartpower_set_mode_for_analysis -mode {value}
```

Arguments

`-mode {value}`

Specifies the mode for cycle-accurate power analysis.

Value	Description
Active	The operating mode is set to Active
Standby	The operating mode is set to Standby
Flash*Freeze	The operating mode is set to Flash*Freeze

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

The following example sets the mode for analysis to active:

```
smartpower_set_mode_for_analysis -mode {active}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_set_mode_for_pdpr

This SmartPower Tcl command sets the operating mode used by the Power Driven Place and Route (PDPR) tool during power optimization.

```
smartpower_set_mode_for_pdpr -opmode { value}
```

Parameters

`-opmode {value}`

Value must be a valid operating mode.

This parameter is mandatory.

Sets the operating mode for your power driven place and route.

Exceptions

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Return Value

This command does not return a value.

Examples

This example sets the Active mode as the operating mode for Power Driven Place and Route.

```
set_mode_for_pdpr -opmode "Active"
```

This example creates a custom mode and set it to be used by Power Driven Place and Route (PDPR).

```
smartpower_add_new_custom_mode –name “MyCustomMode” \
–description “for PDPR” –base_mode “Active”
smartpower_set_mode_for_pdpr –opmode “MyCustomMode
```

See Also

Tcl Command Documentation Conventions

smartpower_set_operating_condition

Tcl command; sets the operating conditions used in SmartPower to one of the pre-defined types.

```
smartpower_set_operating_condition -opcond {value}
```

Arguments

`-opcond {value}`

Specifies the value of the operating condition. The following table shows the acceptable values for this argument:

Value	Description
best	Sets the operating conditions to best
typical	Sets the operating conditions to typical
worst	Sets the operating conditions to worst

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example sets the operating conditions to best:

```
smartpower_set_operating_condition -opcond {best}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

This command was obsoleted in SmartPower v8.5. Update your script to use `smartpower_set_pin_probability` to set the pin probability.

Note: The information below is obsolete and should only be used as reference when executing previously-created scripts. Update your scripts to use smartpower_set_pin_probability.

Enables you to set the probability value of a pin driving an enable pin. For I/Os, if you do not use this command, the probability of the IOEnableSet is used. For memories, if you do not use this command, the probability of the MemoriesEnableSet is used.

```
smartpower_set_pin_enable_rate -pin_name {pin_name} –pin_enable_rate {value}
```

Arguments

`-pin_name {pin_name}`

Specifies the name of a pin for which the probability will be set. This pin must be the direct driver of an enable pin.

`-pin_enable_rate {value}`

Specifies the value of the pin probability as a percentage, which can be any positive decimal between 0 and 100, inclusive.

Supported Families

SmartFusion, IGLOO, ProASIC3, Fusion

Exceptions

None

Examples

The following example sets the probability of the pin driving the enable pin of a bidirectional I/O

```
smartpower_set_pin_enable_rate -pin_name mybibuf/U0/U1:EOUT   \
–pin_enable_rate 50.4
```

smartpower_set_pin_frequency

Tcl command; sets the frequency of a pin in megahertz (MHz). If you do not use this command, each pin will have default frequency based on its domain.

```
smartpower_set_pin_frequency -pin_name {pin_name} -pin_freq {value}
```

Arguments

`-pin_name {pin_name}`

Specifies the name of the pin for which the frequency will be set.

`-pin_freq {value}`

Specifies the value of the frequency in MHz, which can be any positive decimal number.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

The *pin_name* must be the name of a pin that already exists in the design and already belongs to a domain.

When specifying the unit, a space must be between the frequency value and the unit.

Examples

This example sets the frequency of the pin named "count8_clock" to 100 MHz:

```
smartpower_set_pin_frequency -pin_name {count8_clock} -pin_freq {100}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

`smartpower_remove_pin_frequency`

smartpower_set_preference

Tcl command; sets the following preferences: power unit, frequency unit, operating mode, operating conditions, and toggle. These preferences can also be set from the preferences dialog box.

```
smartpower_set_preference -powerunit {value} -frequnit {value} -opmode {value} -opcond
{value} -toggle {value}
```

Arguments

`-powerunit {value}`

Specifies the unit in which power is set. The following table shows the acceptable values for this argument:

Value	Description
W	The power unit is set to watts
mW	The power unit is set to milliwatts
uW	The power unit is set to microwatts

`-frequnit {value}`

Specifies the unit in which frequency is set. The following table shows the acceptable values for this argument:

Value	Description
Hz	The frequency unit is set to hertz
kHz	The frequency unit is set to kilohertz
MHz	The frequency unit is set to megahertz

`-opmode {value}`

Specifies the operating mode. The following table shows the acceptable values for this argument:

Value	Description
active	The operating mode is set to active
static	The operating mode is set to static
sleep	The operating mode is set to sleep
Flash*Freeze	The operating mode is set to Flash*Freeze
shutdown	The operating mode is set to shutdown

`-opcond {value}`

Specifies the operating condition. The following table shows the acceptable values for this argument:

Value	Description
worst	The operating condition is set to worst case
typical	The operating condition is set to typical case
best	The operating condition is set to best case

`-toggle {value}`

Specifies the toggle. The following table shows the acceptable values for this argument:

Value	Description
true	The toggle is set to true
false	The toggle is set to false

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

- The following arguments have been removed. Running the script will trigger a warning message: Warning: Invalid argument: -argname "argvalue" Ignored. Ignore the warning.

-maxblocks {integer > 0}

-maxpins [{integer > 0}

-sortorder {ascending, descending}

-sortby {powervalues, alphabetical}

- Flash*Freeze, Sleep, and Shutdown are available only for certain families and devices.
- Worst and Best operating conditions are available only for certain families and devices.

Examples

This example sets the frequency of the power unit to "watts", the frequency unit to "Hz", the operating mode to "active", the operating condition to "typical", and the toggle to "true":

```
smartpower_set_setpreferences -powerunit {w} -frequnit {hz} -opmode {active} -opcond {typical} -toggle {true}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

SmartPower Preferences

smartpower_set_scenario_for_analysis

Tcl command; sets the scenario for cycle-accurate power analysis.

```
smartpower_set_scenario_for_analysis -scenario{value}
```

Arguments

`-scenario {value}`

Specifies the mode for cycle-accurate power analysis.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

The following example sets the scenario for analysis to my_scenario:

```
smartpower_set_scenario_for_analysis -scenario {my_scenario}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_set_process

Tcl command; sets the process used in SmartPower to one of the pre-defined types.

```
smartpower_set_process -process {value}
```

Arguments

`-process {value}`

Specifies the value of the operating condition. The following table shows the acceptable values for this argument:

Value	Description
Typical	Sets the process for SmartPower to typical
Maximum	Sets the process for SmartPower to maximum

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example sets the operating conditions to typical:

```
smartpower_set_process -process {Typical}
```

See Also

Tcl documentation conventions

smartpower_set_temperature_opcond

Tcl command; sets the temperature in the operating conditions to one of the pre-defined types.

```
smartpower_set_temperature_opcond -use{value}
```

Arguments

`-use{value}`

Specifies the temperature in the operating conditions. The following table shows the acceptable values for this argument:

Value	Description
oprange	Sets the temperature in the operating conditions as specified in your Project Settings.
design	Sets the temperature in the operating conditions as specified in the SmartPower design-wide operating range. Applies to SmartPower only.
mode	Sets the temperature in the operating conditions as specified in the SmartPower mode-specific operating range. Applies to SmartPower only.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example sets the temperature in the operating conditions as specified in the custom mode-settings:

```
smartpower_set_temperature_opcond -use{mode}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_set_thermalmode

Tcl command; sets the mode of computing junction temperature.

```
smartpower_set_thermalmode -mode {value}
```

Arguments

`-mode {value}`

Specifies the mode in which the junction temperature is computed. The following table shows the acceptable values for this argument:

Value	Description
ambient	The junction temperature will be iteratively computed with total static power
opcond	The junction temperature will be given as one of the operating condition range values specified in the device selection

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Notes

To compute the junction temperature, set the following three commands: smartpower_set_thermalmode, smartpower_set_tambient and smartpower_set_cooling. The junction temperature will be updated when an output command is executed, such as report(Power).

Examples

The following example sets the computing of the junction temperature to ambient mode:

```
smartpower_set_thermalmode -mode {ambient}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_set_voltage_opcond

Tcl command; sets the voltage in the operating conditions.

```
smartpower_set_voltage_opcond -voltage{value} -use{value}
```

Arguments

`-voltage{value}`

Specifies the voltage supply in the operating conditions. The following table shows the acceptable values for this argument:

Value	Description

Value	Description
VDD	Sets the voltage operating conditions for VDD
VDDI 2.5	Sets the voltage operating conditions fo r VDDI 2.5
VPP	Sets the voltage operating conditions for VPP

`-use{value}`

Specifies the voltage in the operating conditions for each voltage supply. The following table shows the acceptable values for this argument:

Value	Description
oprange	Sets the voltage in the operating conditions as specified in your Project Settings.
design	Sets the voltage in the operating conditions as specified in the SmartPower design-wide operating range. Applies to SmartPower only.
mode	Sets the voltage in the operating conditions as specified in the SmartPower mode-specific operating range. Applies to SmartPower only.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example sets the VCCA as specified in the SmartPower mode-specific settings:

```
smartpower_set_voltage_opcond -voltage{vcca} -use{mode}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_temperature_opcond_set_design_wide

Tcl command; sets the temperature for SmartPower design-wide operating conditions.

```
smartpower_temperature_opcond_set_design_wide -best{value} -typical{value} -worst{value} -
thermal_mode{value}
```

Arguments

`-best{value}`

Specifies the best temperature (in degrees Celsius) used for design-wide operating conditions.

`-typical{value}`

Specifies the typical temperature (in degrees Celsius) used for design-wide operating conditions.

`-worst{value}`

Specifies the worst temperature (in degrees Celsius) used for design-wide operating conditions.

`-thermal_mode{value}`

Specifies the mode in which the junction temperature is computed. The following table shows the acceptable values for this argument:

Value	Description
ambient	The junction temperature will be iteratively computed with total static power
opcond	The junction temperature will be given as one of the operating condition range values specified in the device selection

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example sets the temperature for design-wide operating conditions to Best 20, Typical 30, and Worst 60:

```
smartpower_temperature_opcond_set_design_wide -best{20} -typical{30} -worst{60}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_temperature_opcond_set_mode_specific

Tcl command; sets the temperature for SmartPower mode-specific operating conditions.

```
smartpower_temperature_opcond_set_mode_specific -opmode{value} -thermal_mode{value} -
best{value} -typical{value} -worst{value} -thermal_mode{value}
```

Arguments

`-opmode {value}`

Specifies the operating mode. The following table shows the acceptable values for this argument:

Value	Description
Active	The operating mode is set to Active
Standby	The operating mode is set to Standby
Flash*Freeze	The operating mode is set to Flash*Freeze

`-thermal_mode{value}`

Specifies the mode in which the junction temperature is computed. The following table shows the acceptable values for this argument:

Value	Description
ambient	The junction temperature will be iteratively computed with total static power

Value	Description
opcond	The junction temperature will be given as one of the operating condition range values specified in the device selection

`-best{value}`

Specifies the best temperature (in degrees Celsius) for the selected mode.

`-typical{value}`

Specifies the typical temperature (in degrees Celsius) for the selected mode.

`-worst{value}`

Specifies the worst temperature (in degrees Celsius) for the selected mode.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example sets the temperature for mode-specific operating conditions for mode1:

```
smartpower_temperature_opcond_set_mode_specific -mode{mode1} -best{20} -typical{30} -worst{60}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_voltage_opcond_set_design_wide

Tcl command; sets the voltage settings for SmartPower design-wide operating conditions.

```
smartpower_voltage_opcond_set_design_wide -voltage{value} -best{value} -typical{value} -worst{value}
```

Arguments

`-voltage{value}`

Specifies the voltage supply in the operating conditions. The following table shows the acceptable values for this argument:

Value	Description
VDD	Sets the voltage operating conditions for VDD
VDDI 2.5	Sets the voltage operating conditions for VDDI 2.5
VPP	Sets the voltage operating conditions for VPP
VCCA	Sets the voltage operating conditions for VCCA
VCCI 3.3	Sets the voltage operating conditions for VCCI 3.3
VCCI 2.5	Sets the voltage operating conditions for VCCI 2.5
VCCI 1.8	Sets the voltage operating conditions for VCCI 1.8

Value	Description
VCCI 1.5	Sets the voltage operating conditions for VCCI 1.5
VCC33A	Sets the voltage operating conditions for VCC33A
VCCDA	Sets the voltage operating conditions for VCCDA

`-best{value}`

Specifies the best voltage used for design-wide operating conditions.

`-typical{value}`

Specifies the typical voltage used for design-wide operating conditions.

`-worst{value}`

Specifies the worst voltage used for design-wide operating conditions.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example sets VCCA for design-wide to best 20, typical 30 and worst 40:

```
smartpower_voltage_opcond_set_design_wide -voltage{VCCA} -best{20} -typical{30} -
worst{40}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

smartpower_voltage_opcond_set_mode_specific

Tcl command; sets the voltage settings for SmartPower mode-specific use operating conditions.

```
smartpower_voltage_opcond_set_mode_specific -opmode{value} -voltage{value} -best{value} -
typical{value} -worst{value}
```

Arguments

`-opmode {value}`

Use this option to specify the mode from which the operating conditions are extracted to generate the report.

Value	Description
Active	The operating mode is set to Active
Standby	The operating mode is set to Standby
Flash*Freeze	The operating mode is set to Flash*Freeze

`-voltage{value}`

Specifies the voltage in the operating conditions. The following table shows the acceptable values for this argument:

Value	Description
VDD	Sets the voltage operating conditions for VDD
VDDI 2.5	Sets the voltage operating conditions for VDDI 2.5
VPP	Sets the voltage operating conditions for VPP
VCCA	Sets the voltage operating conditions for VCCA
VCCI 3.3	Sets the voltage operating conditions for VCCI 3.3
VCCI 2.5	Sets the voltage operating conditions for VCCI 2.5
VCCI 1.8	Sets the voltage operating conditions for VCCI 1.8
VCCI 1.5	Sets the voltage operating conditions for VCCI 1.5
VCC33A	Sets the voltage operating conditions for VCC33A
VCCDA	Sets the voltage operating conditions for VCCDA

`-best{value}`

Specifies the best voltage used for mode-specific operating conditions.

`-typical{value}`

Specifies the typical voltage used for mode-specific operating conditions.

`-worst{value}`

Specifies the worst voltage used for mode-specific operating conditions.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example sets the voltage for the static mode and sets best to 20, typical to 30 and worst to 40:

```
smartpower_voltage_opcond_set_mode_specific -opmode{active} -voltage{VCCA} -best{20} -typical{30} -worst{40}
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

st_commit

Tcl command; saves the changes made in SmartTime to the design (.adb) file

```
st_commit
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
st_commit
```

See Also

st_restore

Tcl documentation conventions

Designer Tcl Command Reference

st_create_set

Tcl command; creates a set of paths to be analyzed. Use the arguments to specify which paths to include. To create a set that is a subset of a clock domain, specify it with `-clock` and `-type`. To create a set that is a subset of an inter-clock domain set, specify it with `-source_clock` and `-sink_clock`. To create a set that is a subset (filter) of an existing named set, specify the set to be filtered with `-from_set`.

To create a set that is not derived from an existing set, you must provide both the `-source` *`pin_list`* and `-sink`*`pin_list`* derived. Otherwise, the `-source` and `-sink` arguments act as filters on the pins from the parent set. You must give each new set a unique name in the design.

```
st_create_set -name name
[-parent_set name ]
[-clockclock_id -type value ]
[-in_to_out]
[-source_clock clock_id -sink_clock clock_id]
[-source pin_list ] -sink pin_list ]
```

Arguments

`-name` *`name`*

Specifies a unique name for the newly create path set.

-parent_set *`name`*

Specifies the name of the set to filter.

-clock *`clock_id`*

Specifies that the set is to be a subset of the given clock domain. This argument is valid only if you also specify the -type argument.

`-type` *`value`*

Specifies the predefined set type on which to base the new path set. You can only use this argument with the -clock argument, not by itself.

Value	Description
reg_to_reg	Paths between registers in the design
async_to_reg	Paths from asynchronous pins to registers
reg_to_async	Paths from registers to asynchronous pins
external_recovery	The set of paths from inputs to asynchronous pins
external_removal	The set of paths from inputs to asynchronous pins
external_setup	Paths from input ports to registers
external_hold	Paths from input ports to registers
clock_to_out	Paths from registers to output ports

`-in_to_out`

Specifies that the set is based on the “Input to Output” set, which includes paths that start at input ports and end at output ports.

`-source_clock` *`clock_id`*

Specifies that the set will be a subset of an inter-clock domain set with the given source clock.

You can only use this option with the -sink_clock option, not by itself.

`-sink_clock` *`clock_id`*

Specifies that the set will be a subset of an inter-clock domain set with the given sink clock.

You can only use this option with the -source_clock option, not by itself.

`-source` *`pin_list`*

Specifies a filter on the source pins of the parent set. If you do not specify a parent set, this option filters all pins in the current design.

`-sink` *`pin_list`*

Specifies a filter on the sink pins of the parent set. If you do not specify a parent set, this option filters all pins in the current design.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
st_create_set –name { my_user_set } –source { C* } –sink { D* }
st_create_set –name { my_other_user_set } –from_set { my_user_set } –source { CL* }
st_create_set –name { adder } –clock { ALU_CLOCK } –type { REG_TO_REG }  -sink { ADDER* }
st_create_set –name { another_set } –source_clock { EXTERN_CLOCK } –sink_clock { MY_GEN_CLOCK }
st_create_set –name { some_p2p } –pin2pin –to { T* }
```

See Also

Designer Tcl Command Reference

Tcl documentation conventions

`st_remove_set`

st_edit_set

Tcl command; modifies the paths in a user set.

```
st_edit_set -name name
 [-source pin_list ] [-sink  pin_list ]
 [-rename_to name ]
```

Arguments

`-name` *`name`*

Specifies the name of the set to modify.

`-source` *`pin_list`*

If the set is a subset of another set, specifies a filter on the source pins from the parent set. Otherwise, this option specifies the source pins of the set.

`-sink` *`pin_list`*

If the set is a subset of another set, specifies a filter on the sink pins from the parent set. Otherwise, this option specifies the sink pins of the set.

`-rename_to` *name*

Specifies a new name for the set.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
st_edit_set –name { my_user_set} –rename_to { my_critical_pins }
st_edit_set –name { adder } –sink { ADD* }
```

See Also

Designer Tcl Command Reference

Tcl documentation conventions

st_create_set

st_remove_set

st_expand_path

Tcl command; displays expanded path information (path details) for paths. The paths to be expanded are identified by the parameters required to display these paths with st_list_paths. For example, to expand the first path listed with st_list_paths -clock {MYCLOCK} -type {register_to_register}, use the command st_expand_path -clock {MYCLOCK} -type {register_to_register}. Path details contain the pin name, type, net name, cell name, operation, delay, total delay, and edge as well as the arrival time, required time, and slack. These details are the same as details available in the SmartTime Expanded Path window.

```
st_expand_path [-set name]
[-clock clock_id -type value]
[-in_to_out]
[-source_clock clock_id -sink_clock clock_id]
[-source pin_list] [-sink pin_list]
[-analysis value]
[-index list_of_indices]
[-format value]
```

Arguments

-set *name*

Displays a list of paths from the named set. You can either use the -set option to specify a set name, or use both -clock and -type to specify a set. A list of valid set names includes "in_to_out", as well as any user set names.

-clock *clock_id*

Displays the set of paths belonging to the specified clock domain. You can either use this option along with -type to specify a set or use the -set option to specify the name of the set to display.

`-in_to_out`

Specifies that the paths should be from the set "Input to Output, which includes paths that start at input ports and end at output ports.

`-type` *value*

Specifies the type of paths in the clock domain to display in a list. You can only use this option with the -clock option, not by itself. You can either use this option along with -clock to specify a set or use the -set option to specify a set name.

Value	Description
reg_to_reg	Paths between registers in the design

Value	Description
async_to_reg	Paths from asynchronous pins to registers
reg_to_asyn	Paths from registers to asynchronous pins
external_recovery	The set of paths from inputs to asynchronous pins
external_removal	The set of paths from inputs to asynchronous pins
external_setup	Paths from input ports to registers
	Paths from input ports to registers
clock_to_out	Paths from registers to output ports

`-source_clock` *`clock_id`*

Displays a list of timing paths for an inter-clock domain set belonging to the source clock specified. You can only use this option with the -sink_clock option, not by itself.

`-sink_clock` *`clock_id`*

Displays a list of timing paths for an inter-clock domain set belonging to the sink clock specified. You can only use this option with the -source_clock option, not by itself.

`-source` *`pin_list`*

Specifies a filter on the source pins of the paths to be listed.

`-sink` *`pin_list`*

Specifies a filter on the sink pins of the paths to be listed.

`-analysis` *`name`*

Specifies the analysis type for the paths to be listed. The following table shows the acceptable values for this argument:

Value	Description
maxdelay	Maximum delay analysis
mindelay	Minimum delay analysis

`-index` *`list_of_indices`*

Specifies which paths to display. The index starts at 1 and defaults to 1. Only values lower than the max_paths option will be expanded.

`-format` *`value`*

Specifies the file format of the output. The following table shows the acceptable values for this argument:

Value	Description
text	ASCII text format
csv	Comma separated value fie format

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

Note: The following example returns a list of five paths:

```
st_expand_path –clock { myclock } –type {reg_to_reg }
st_expand_path –clock {myclock} –type {reg_to_reg} –index { 1 2 3 } –format text
```

See Also

Designer Tcl Command Reference

Tcl documentation conventions

st_list_paths

st_list_paths

Tcl command; displays the list of paths in the same tabular format shown in SmartTime.

```
st_list_paths [-set name ]
[-clock clock_id -type value ]
 [-in_to_out]
 [-source_clock clock_id -sink_clock  clock_id]
 [-source pin_list ] [-sink pin_list ]
 [-analysis value ]
 [-format value ]
```

Arguments

`-set name`

Displays a list of paths from the named set. You can either use the –set option to specify a set name, or use both –clock and –type to specify a set. A list of valid set names includes “in_to_out”, as well as any user set names.

-clock *clock_id*

Displays the set of paths belonging to the specified clock domain. You can either use this option along with -type to specify a set or use the -set option to specify the name of the set to display.

`-in_to_out`

Specifies that the paths should be from the set “Input to Output”, which includes paths that start at input ports and end at output ports.

`-type value`

Specifies the type of paths in the clock domain to display in a list. You can only use this option with the -clock option, not by itself. You can either use this option along with -clock to specify a set or use the -set option to specify a set name.

Value	Description
reg_to_reg	Paths between registers in the design
async_to_reg	Paths from asynchronous pins to registers
reg_to_asyn	Paths from registers to asynchronous pins
external_recovery	The set of paths from inputs to asynchronous pins
external_removal	The set of paths from inputs to asynchronous pins
external_setup	Paths from input ports to registers
	Paths from input ports to registers

Value	Description
clock_to_out	Paths from registers to output ports

`-source_clock` **`clock_id`**

Displays a list of timing paths for an inter-clock domain set belonging to the source clock specified. You can only use this option with the -sink_clock option, not by itself.

`-sink_clock` **`clock_id`**

Displays a list of timing paths for an inter-clock domain set belonging to the sink clock specified. You can only use this option with the -source_clock option, not by itself.

`-source` **`pin_list`**

Specifies a filter on the source pins of the paths to be listed.

`-sink` **`pin_list`**

Specifies a filter on the sink pins of the paths to be listed.

`-analysis` *`name`*

Specifies the analysis type for the paths to be listed. The following table shows the acceptable values for this argument:

Value	Description
maxdelay	Maximum delay analysis
mindelay	Minimum delay analysis

`-format` *`value`*

Specifies the file format of the output. The following table shows the acceptable values for this argument:

Value	Description
text	ASCII text format
csv	Comma separated value fie format

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
st_list_paths –set  { myset }
st_list_paths –analysis mindelay –clock { myclock } –type { reg_to_reg } –format csv
```

The list of paths can be written to a file with the following Tcl commands:

```
set outfile [ open “pathlisting.csv” w]
puts $outfile [ st_list_paths –format csv –set { myset} ]
close $outfile
```

See Also

Designer Tcl Command Reference

Tcl documentation conventions

st_expand_path

st_remove_set

Tcl command; deletes a user set from the design.

```
st_remove_set -name name
```

Arguments

`-name name`

Specifies the name of the set to delete.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
st_remove_set { clockset1 }
```

See Also

Designer Tcl Command Reference

Tcl documentation conventions

st_create_set

st_restore

Tcl command; restores constraints previously committed in SmartTime.

```
st_restore
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
st_restore
```

See Also

st_commit

Tcl documentation conventions

Designer Tcl Command Reference

st_set_options (SmartFusion, IGLOO, ProASIC3, Fusion only)

Tcl command; sets options for timing analysis. With no parameters given, it will display the current settings of the options. For SmartFusion, IGLOO, ProASIC3, Fusion families, these options also affect timing-driven place-and-route.

```
st_set_options [-max_opcond value ]
 [-min_opcond value]
 [-interclockdomain_analysis value]
 [-use_bibuf_loopbacks value]
```

```
[-enable_recovery_removal_checks value]
[-break_at_async value]
[-filter_when_slack_below value]
[-filter_when_slack_above value]
[-remove_slack_filters]
[-limit_max_paths value]
[-expand_clock_network value]
[-expand_parallel_paths value]
[-analysis_scenario value]
[-tdpr_scenario value]
[-reset]
```

Arguments

-max_opcond *value*

Sets the operating condition to use for Maximum Delay Analysis. The following table shows the acceptable values for this argument:

Value	Description
worst	Use Worst Case conditions for Maximum Delay Analysis
typ	Use Typical conditions for Maximum Delay Analysis
best	Use Best Case conditions for Maximum Delay Analysis

-min_opcond *value*

Sets the operating condition to use for Minimum Delay Analysis. The following table shows the acceptable values for this argument:

Value	Description
best	Use Best Case conditions for Minimum Delay Analysis
typ	Use Typical conditions for Minimum Delay Analysis
worst	Use Worst Case conditions for Minimum Delay Analysis

-interclockdomain_analysis *value*

Enables or disables inter-clock domain analysis.

Value	Description
yes	Enables inter-clock domain analysis
no	Disables inter-clock domain analysis

-use_bibuf_loopbacks *value*

Enables or disables loopback in bibufs.

Value	Description
yes	Enables loopback in bibufs
no	Disables loopback in bibufs

-enable_recovery_removal_checks *value*

Enables or disables recovery and removal checks.

Value	Description
yes	Enables recovery and removal checks
no	Disables recovery and removal checks

-break_at_async *value*

Enables or disables breaking paths at asynchronous ports.

Value	Description
yes	Enables breaking paths at asynchronous ports
no	Disables breaking paths at asynchronous ports

-filter_when_slack_below *value*

Do not show paths with slack below x.

-filter_when_slack_above *value*

Do not show paths with slack above y.

-remove_slack_filters

Remove all existing slack filters.

-limit_max_paths *value*

Limit path reporting commands to a maximum of <n> paths, where n is a value of 0 or higher.

-expand_clock_network *value*

Enables or disables expanded clock network information in expanded paths.

Value	Description
yes	Enables expanded clock network information in paths
no	Disables expanded clock network information in paths

-expand_parallel_paths *value*

Expand a maximum of <n> parallel paths, where n is a value of 0 or higher. If n is 0 or 1, only one path will be expanded when viewing expanded paths.

-analysis_scenario *value*

Set the timing constraints scenario to be used for both maximum delay and minimum delay analysis. The argument must be a valid scenario name.

Note: This option does not affect the timing scenario used for TDPR.

-tdpr_scenario *value*

Set the timing constraints scenario to be used by the place and route engine. The argument must be a valid scenario name.

Note: This option does not affect the timing scenario used for analysis.

-reset

Reset all options to their default values, except for scenarios used for analysis and TDPR that remain unchanged.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
st_set_options –max_opcond worst
-min_opcond best
-interclockdomain_analysis true
-enable_removal_recovery_checks true
st_set_options –limit_max_paths 50 –remove_slack_filters
–filter_when_slack_above 3
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

timer_get_path

Tcl command; displays the path between the specified pins in the Log window.

```
timer_get_path -from source_pin -to destination_pin
[-exp value]\
[-sort value]\
[-order value]\
[-case value]\
[-maxpath maximum_paths]\
[-maxexpath maximum_paths_to_expand]\
[-mindelay minimum_delay]\
[-maxdelay maximum_delay]\
[-breakatclk value]\
[-breakatclr value]
```

Arguments

`-from source_pin`

Specifies the name of the source pin for the path.

`-to destination_pin`

Specifies the name of the destination pin for the path.

`-exp value`

Specifies whether to expand the path. The following table shows the acceptable values for this argument:

Value	Description
yes	Expands the path
no	Does not expand the path

`-sort value`

Specifies whether to sort the path by either the actual delay or slack value. The following table shows the acceptable values for this argument:

Value	Description
actual	Sorts the path by the actual delay value

Value	Description
slack	Sorts the path by the slack value

`-order` *value*

Specifies whether the list is based on maximum or minimum delay analysis. The following table shows the acceptable values for this argument:

Value	Description
long	The paths are listed based on the maximum delay analysis
short	The paths are listed based on the minimum delay analysis

`-case` *value*

Specifies whether the report will include the worst, typical, or best case timing numbers. The following table shows the acceptable values for this argument:

Value	Description
worst	Includes worst case timing numbers
typ	Includes typical case timing numbers
best	Includes best case timing numbers

`-maxpath` *maximum_paths*

Specifies the maximum number of paths to display.

`-maxexpath` *maximum_paths_to_expand*

Specifies the maximum number of paths to expand.

`-mindelay` *minimum_delay*

Specifies the path delay in the minimum delay analysis mode.

`-maxdelay` *maximum_delay*

Specifies the path delay in the maximum delay analysis mode.

`-breakatclk` *value*

Specifies whether to break the paths at the register clock pins. The following table shows the acceptable values for this argument:

Value	Description
yes	Breaks the paths at the register clock pins
no	Does not break the paths at the register clock pins

`-breakatclr` *value*

Specifies whether to break the paths at the register clear pins. The following table shows the acceptable values for this argument:

Value	Description
yes	Breaks the paths at the register clear pins

Value	Description
no	Does not break the paths at the register clear pins

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

The following example returns the paths from input port headdr_dat<31> to the input pin of register u0_headdr_data1_reg/data_out_t_31 under typical conditions.

```
timer_get_path -from "headdr_dat<31>" \
-to  "u0_headdr_data1_reg/data_out_t_31/U0:D" \
-case typ  \
-exp "yes" \
-maxpath "100" \
-maxexpapth "10"
```

The following example returns the paths from the clock pin of register gearbox_inst/bits64_out_reg<55> to the output port pma_tx_data_64bit[55]

```
timer_get_path -from "gearbox_inst/bits64_out_reg<55>/U0:CLK" \
     -to {pma_tx_data_64bit[55]} \
     -exp "yes"
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

timer_get_clock_actuals

Tcl command; displays the actual clock frequency in the Log window, when the timing analysis tool is initiated.

```
timer_get_clock_actuals -clock clock_name
```

Arguments

-clock *clock_name*

Specifies the name of the clock with the frequency (or period) to display.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example displays the actual clock frequency of clock clk1 in the Log window:

```
timer_get_clock_actuals -clock clk1
```

See Also

timer_get_clock_constraints

Tcl documentation conventions

Designer Tcl Command Reference

timer_get_clock_constraints

Tcl command; returns the constraints (period, frequency, and duty cycle) on the specified clock.

```
timer_get_clock_constraints -clock clock_name
```

Arguments

-clock *clock_name*

Specifies the name of the clock with the constraint to display.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

The following example displays the clock constraints on the clock clk in the Log window:

```
timer_get_clock_constraints -clock clk
```

See Also

timer_get_clock_actuals

Tcl documentation conventions

Designer Tcl Command Reference

timer_get_maxdelay

Tcl command; displays the maximum delay constraint between two pins in a path in the Log window.

```
timer_get_maxdelay -from source_pin -to destination_pin
```

Arguments

-from *source_pin*

Specifies the name of the source pin in the path.

-to *destination_pin*

Specifies the name of the destination pin in the path.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

The following example displays the maximum delay constraint from the pin clk166 to the pin reg_q_a_9_/U0:CLK in the Log window:

```
timer_get_maxdelay -from {clk166} -to {reg_q_a_9_/U0:CLK}
```

See Also

timer_set_maxdelay

Tcl documentation conventions

Designer Tcl Command Reference

timer_get_path_constraints

Tcl command; displays the path constraints that were set as the maximum delay constraint in the timing analysis tool.

```
timer_get_path_constraints
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

This command lists the paths constrained by maximum delay values. The information is displayed in the Log window. If no maximum delay constraints are set, this command does not report anything.

Examples

Invoking timer_get_path_constraints displays the following paths and their delay constraints in the Log window:

```
max_delay -from [all_inputs] -to [all_outputs] = 12 ns
max_delay -from p_f_testwdata0 p_f_testwdata1 -to p_f_dacuwdata0 p_f_dacuwdata1
r_f_dacuwdata0 r_f_dacuwdata1 = 8 ns
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

timer_remove_stop

Tcl command; removes the previously entered path stop constraint on the specified pin.

```
timer_remove_stop -pin pin_name
```

Arguments

`-pin pin_name`

Specifies the name of the pin from which to remove the path stop constraint.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

If you remove a path stop constraint using the Timer GUI, and then export a script using **File > Export** > **Script files**, the resulting script will contain `timer_remove_pass -pin pin_name` instead of `timer_remove_stop -pin pin_name`.

Exceptions

- For the SmartFusion2, SmartFusion, IGLOO, ProASIC3, Fusion families, best practice is to use the following flow:
    1. Open **SmartTime** > Set False Path Constraint dialog box.

2. Look for the pin name in the **Through:** list (Note: You must not have any entry selected in the **From** or **To** lists).
3. Delete this pin.

Examples

The following example removes the path stop constraint on the clear pin reg_q_a_0_:CLR:

```
timer_remove_stop -pin {reg_q_a_0_:CLR}
```

See Also

Tcl documentation conventions

Set False Path Constraint dialog box

Designer Tcl Command Reference

timer_remove_all_constraints

Tcl command; removes all timing constraints in the current design.

```
timer_remove_all_constraints
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

The following example removes all of the constraints from the design and then commits the changes:

```
timer_remove_all_constraints
timer_commit
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

timer_restore

Tcl command; restores constraints previously committed in Timer.

```
timer_restore
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
timer_restore
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

SmartFusion, IGLOO, ProASIC3, and Fusion - SmartTime

all_registers

Tcl command; returns an object representing register pins or cells in the current scenario based on the given parameters.

```
all_registers [-clock clock_name]
[-async_pins][-output_pins][-data_pins][-clock_pins]
```

Arguments

-clock *clock_name*

Specifies the name of the clock domain to which the registers belong. If no clock is specified, all registers in the design will be targeted.

-async_pins

Lists all register pins that are async pins for the specified clock (or all registers asynchronous pins in the design).

-output_pins

Lists all register pins that are output pins for the specified clock (or all registers output pins in the design).

-data_pins

Lists all register pins that are data pins for the specified clock (or all registers data pins in the design).

-clock_pins

Lists all register pins that are data pins for the specified clock (or all registers clock pins in the design).

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Exceptions

You can only use this command as part of a –from, -to, or –through argument in the following Tcl commands: set_min_delay, set_max_delay, set_multicycle_path, and set_false_path.

Examples

```
set_max_delay 2.000 -from { ff_m:CLK ff_s2:CLK } -to [all_registers -clock_pins -clock {
ff_m:Q }]
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

check_timing_constraints

Tcl command; checks all timing constraints in the current timing scenario for validity.

```
check_timing_constraints
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
check_timing_constraints
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

clone_scenario

Tcl command; creates a new timing scenario by duplicating an existing one. You must provide a unique name (that is, it cannot already be used by another timing scenario).

```
clone_scenario name -source origin
```

Arguments

name

Specifies the name of the new timing scenario to create.

`-source` *origin*

Specifies the source of the timing scenario to clone (copy). The source must be a valid, existing timing scenario.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

This command creates a timing scenario with the specified name, which includes a copy of all constraints in the original scenario (specified with the -source parameter). The new scenario is then added to the list of scenarios.

Example

```
clone_scenario scenario_A –source {Primary}
```

See Also

`create_scenario`

`delete_scenario`

Tcl documentation conventions

Designer Tcl Command Reference

create_clock

Tcl command; creates a clock constraint on the specified ports/pins, or a virtual clock if no source other than a name is specified.

```
create_clock -period period_value [-name clock_name]
[-waveform> edge_list][source_objects]
```

Arguments

`-period` *`period_value`*

Specifies the clock period in nanoseconds. The value you specify is the minimum time over which the clock waveform repeats. The period_value must be greater than zero.

`-name` *`clock_name`*

Specifies the name of the clock constraint. You must specify either a clock name or a source.

`-waveform` *`edge_list`*

Specifies the rise and fall times of the clock waveform in ns over a complete clock period. There must be exactly two transitions in the list, a rising transition followed by a falling transition. You can define a clock starting with a falling edge by providing an edge list where fall time is less than rise time. If you do not specify -waveform option, the tool creates a default waveform, with a rising edge at instant 0.0 ns and a falling edge at instant (period_value/2)ns.

`source_objects`

Specifies the source of the clock constraint. The source can be ports, pins, or nets in the design. If you specify a clock constraint on a pin that already has a clock, the new clock replaces the existing one. You must specify either a source or a clock name.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Creates a clock in the current design at the declared source and defines its period and waveform. The static timing analysis tool uses this information to propagate the waveform across the clock network to the clock pins of all sequential elements driven by this clock source.

The clock information is also used to compute the slacks in the specified clock domain that drive optimization tools such as place-and-route.

Examples

The following example creates two clocks on ports CK1 and CK2 with a period of 6, a rising edge at 0, and a falling edge at 3:

```
create_clock -name {my_user_clock} -period 6 CK1
create_clock -name {my_other_user_clock} –period 6 –waveform {0 3} {CK2}
```

The following example creates a clock on port CK3 with a period of 7, a rising edge at 2, and a falling edge at 4:

```
create_clock –period 7 –waveform {2 4} [get_ports {CK3}]
```

See Also

create_generated_clock

Tcl Command Documentation Conventions

Designer Tcl Command Reference

create_generated_clock

Tcl command; creates an internally generated clock constraint on the ports/pins and defines its characteristics.

```
create_generated_clock [-name name] -source reference_pin [-divide_by divide_factor] [-
multiply_by multiply_factor] [-invert] source
```

Arguments

`-name` *`name`*

Specifies the name of the clock constraint.

`-source` *`reference_pin`*

Specifies the reference pin in the design from which the clock waveform is to be derived.

`-divide_by` *`divide_factor`*

Specifies the frequency division factor. For instance if the *divide_factor* is equal to 2, the generated clock period is twice the reference clock period.

`-multiply_by` *`multiply_factor`*

Specifies the frequency multiplication factor. For instance if the *multiply_factor* is equal to 2, the generated clock period is half the reference clock period.

`-invert`

Specifies that the generated clock waveform is inverted with respect to the reference clock.

`source`

Specifies the source of the clock constraint on internal pins of the design. If you specify a clock constraint on a pin that already has a clock, the new clock replaces the existing clock. Only one source is accepted. Wildcards are accepted as long as the resolution shows one pin.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Creates a generated clock in the current design at a declared source by defining its frequency with respect to the frequency at the reference pin. The static timing analysis tool uses this information to compute and propagate its waveform across the clock network to the clock pins of all sequential elements driven by this source.

The generated clock information is also used to compute the slacks in the specified clock domain that drive optimization tools such as place-and-route.

Examples

The following example creates a generated clock on pin U1/reg1:Q with a period twice as long as the period at the reference port CLK.

```
create_generated_clock -name {my_user_clock} –divide_by 2 –source [get_ports {CLK}]  U1/reg1:Q
```

The following example creates a generated clock at the primary output of myPLL with a period ¾ of the period at the reference pin clk.

```
create_generated_clock –divide_by 3 –multiply_by 4  -source clk [get_pins {myPLL:CLK1}]
```

See Also

`create_clock`

Tcl Command Documentation Conventions

Designer Tcl Command Reference

create_scenario

Tcl command; creates a new timing scenario with the specified name. You must provide a unique name (that is, it cannot already be used by another timing scenario).

```
create_scenario name
```

Arguments

`name`

Specifies the name of the new timing scenario.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

A timing scenario is a set of timing constraints used with a design. Scenarios enable you to easily refine the set of timing constraints used for Timing-Driven Place-and-Route, so as to achieve timing closure more rapidly.

This command creates an empty timing scenario with the specified name and adds it to the list of scenarios.

Example

```
create_scenario scenario_A
```

See Also

clone_scenario

Tcl Command Documentation Conventions

Designer Tcl Command Reference

delete_scenario

Tcl command; deletes the specified timing scenario.

```
delete_scenario name
```

Arguments

name

Specifies the name of the timing scenario to delete.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

This command deletes the specified timing scenario and all the constraints it contains.

Exceptions

- At least one timing scenario must always be available. If the current scenario is the only one that exists, you cannot delete it.
- Scenarios that are linked to the timing analysis or layout cannot be deleted.

Example

```
delete_scenario scenario_A
```

See Also

create_scenario

Tcl Command Documentation Conventions

Designer Tcl Command Reference

get_cells

Tcl command; returns an object representing the cells (instances) that match those specified in the pattern argument.

```
get_cells pattern
```

Arguments

`pattern`

Specifies the pattern to match the instances to return. For example, "get_cells U18*" returns all instances starting with the characters "U18", where “*” is a wildcard that represents any character string.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

This command returns a collection of instances matching the pattern you specify. You can only use this command as part of a –from, -to, or –through argument in the following Tcl commands: set_max_delay, set_multicycle_path, and set_false_path.

Examples

```
set_max_delay 2 -from [get_cells {reg*}] -to [get_ports {out}]
set_false_path –through [get_cells {Rblock/muxA}]
```

See Also

get_clocks
get_nets
get_pins
get_ports
Tcl Command Documentation Conventions
Designer Tcl Command Reference

get_clocks

Tcl command; returns an object representing the clock(s) that match those specified in the pattern argument in the current timing scenario.

```
get_clocks pattern
```

Arguments

pattern

Specifies the pattern to use to match the clocks set in SmartTime or Timer.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

- If this command is used as a –from argument in either the set maximum (set_max_delay), or set minimum delay (set_min_delay), false path (set_false_path), and multicycle constraints (set_multicycle_path), the clock pins of all the registers related to this clock are used as path start points.

- If this command is used as a –to argument in either the set maximum (set_max_delay), or set minimum delay (set_min_delay), false path (set_false_path), and multicycle constraints (set_multicycle_path), the synchronous pins of all the registers related to this clock are used as path endpoints.

Example

```
set_max_delay -from [get_ports datal] -to \
[get_clocks ck1]
```

See Also

create_clock

create_generated_clock

Tcl Command Documentation Conventions

Designer Tcl Command Reference

get_current_scenario

Tcl command; returns the name of the current timing scenario.

```
get_current_scenario
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
get_current_scenario
```

See Also

set_current_scenario

Tcl documentation conventions

Designer Tcl Command Reference

get_nets

Tcl command; returns an object representing the nets that match those specified in the pattern argument.

```
get_nets pattern
```

Arguments

pattern

Specifies the pattern to match the names of the nets to return. For example, "get_nets N_255*" returns all nets starting with the characters "N_255", where “*” is a wildcard that represents any character string.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

This command returns a collection of nets matching the pattern you specify. You can only use this command as source objects in create clock (create_clock) or create generated clock (create_generated_clock) constraints and as `-through` arguments in the set false path, set minimum delay, set maximum delay, and set multicycle path constraints.

Examples

```
set_max_delay 2 -from [get_ports RDATA1] -through [get_nets {net_chkp1 net_chkqi}]
set_false_path –through [get_nets {Tblk/rm/n*}]
create_clock -name mainCLK -period 2.5 [get_nets {cknet}]
```

See Also

create_clock
create_generated_clock
set_false_path
set_min_delay
set_max_delay
set_multicycle_path
Tcl documentation conventions
Designer Tcl Command Reference

get_pins

Tcl command; returns an object representing the pin(s) that match those specified in the pattern argument.

```
get_pins pattern
```

Arguments

pattern

Specifies the pattern to match the pins to return. For example, "get_pins clock_gen*" returns all pins starting with the characters "clock_gen", where “*” is a wildcard that represents any character string.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
create_clock -period 10 [get_pins clock_gen/reg2:Q]
```

See Also

create_clock
create_generated_clock
set_clock_latency
set_false_path
set_min_delay
set_max_delay
set_multicycle_path
Tcl documentation conventions
Designer Tcl Command Reference

get_ports

Tcl command; returns an object representing the port(s) that match those specified in the pattern argument.

```
get_ports pattern
```

Argument

pattern

Specifies the pattern to match the ports.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
create_clock -period 10 [get_ports CK1]
```

See Also

create_clock
set_clock_latency
set_input_delay
set_output_delay
set_min_delay
set_max_delay
set_false_path
set_multicycle_path
Tcl documentation conventions
Designer Tcl Command Reference

list_clock_latencies

Tcl command; returns details about all of the clock latencies in the current timing constraint scenario.

```
list_clock_latencies
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_clock_latencies]
```

See Also

set_clock_latency
remove_clock_latency
Tcl documentation conventions
Designer Tcl Command Reference

list_clock_uncertainties

Tcl command; returns details about all of the clock uncertainties in the current timing constraint scenario.

```
list_clock_uncertainties
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
list_clock_uncertainties
```

See Also

set_clock_uncertainty
remove_clock_uncertainty
Designer Tcl Command Reference

list_clocks

Tcl command; returns details about all of the clock constraints in the current timing constraint scenario.

```
list_clocks
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_clocks]
```

See Also

create_clock
remove_clock
Tcl documentation conventions
Designer Tcl Command Reference

list_disable_timings

Tcl command; returns the list of disable timing constraints for the current scenario.

```
list_disable_timings
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
list_disable_timings
```

See Also

Designer Tcl Command Reference

list_false_paths

Tcl command; returns details about all of the false paths in the current timing constraint scenario.

```
list_false_paths
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_false_paths]
```

See Also

set_false_path

remove_false_path

Tcl documentation conventions

Designer Tcl Command Reference

list_generated_clocks

Tcl command; returns details about all of the generated clock constraints in the current timing constraint scenario.

```
list_generated_clocks
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_generated_clocks]
```

See Also

create_generated_clock

remove_generated_clock

Tcl documentation conventions

Designer Tcl Command Reference

list_input_delays

Tcl command; returns details about all of the input delay constraints in the current timing constraint scenario.

```
list_input_delays
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_input_delays]
```

See Also

set_input_delay
remove_input_delay
Tcl documentation conventions
Designer Tcl Command Reference

list_max_delays

Tcl command; returns details about all of the maximum delay constraints in the current timing constraint scenario.

```
list_max_delays
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_max_delays]
```

See Also

set_max_delay
remove_max_delay
Tcl documentation conventions
Designer Tcl Command Reference

list_min_delays

Tcl command; returns details about all of the minimum delay constraints in the current timing constraint scenario.

```
list_min_delays
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_min_delays]
```

See Also

set_min_delay
remove_min_delay
Tcl documentation conventions
Designer Tcl Command Reference

list_multicycle_paths

Tcl command; returns details about all of the multicycle paths in the current timing constraint scenario.

```
list_multicycle_paths
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_multicycle_paths]
```

See Also

set_multicycle_path
remove_multicycle_path
Tcl documentation conventions
Designer Tcl Command Reference

list_objects

Tcl command; returns a list of object matching the parameter. Objects can be nets, pins, ports, clocks or instances.

```
list_objects <object>
```

Arguments

Any timing constraint parameter.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following example lists all the inputs in your design:

```
list_objects [all_inputs]
```

You can also use wildcards to filter your list, as in the following command:

```
list_objects [get_ports a*]
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

list_output_delays

Tcl command; returns details about all of the output delay constraints in the current timing constraint scenario.

```
list_output_delays
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_output_delays]
```

See Also

set_output_delay

remove_output_delay

Tcl documentation conventions

Designer Tcl Command Reference

list_scenarios

Tcl command; returns a list of names of all of the available timing scenarios.

```
list_scenarios
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
list_scenarios
```

See Also

get_current_scenario

Tcl documentation conventions

Designer Tcl Command Reference

remove_clock

Tcl command; removes the specified clock constraint from the current timing scenario.

```
remove_clock -name clock_name| -id constraint_ID
```

Arguments

`-name clock_name`

Specifies the name of the clock constraint to remove from the current scenario. You must specify either a clock name or an ID.

`-id constraint_ID`

Specifies the ID of the clock constraint to remove from the current scenario. You must specify either an ID or a clock name that exists in the current scenario.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes the specified clock constraint from the current scenario. If the specified name does not match a clock constraint in the current scenario, or if the specified ID does not refer to a clock constraint, this command fails.

Do not specify both the name and the ID.

Exceptions

You cannot use wildcards when specifying a clock name.

Examples

The following example removes the clock constraint named "my_user_clock":

```
remove_clock -name my_user_clock
```

The following example removes the clock constraint using its ID:

```
set clockId [create_clock –name my_user_clock –period 2]
remove_clock –id $clockId
```

See Also

create_clock

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_clock_latency

Tcl command; removes a clock source latency from the specified clock and from all edges of the clock.

```
remove_clock_latency {-source clock_name_or_source |-id constraint_ID}
```

Arguments

`-source clock_name_or_source`

Specifies either the clock name or source name of the clock constraint from which to remove the clock source latency. You must specify either a clock or source name or its constraint ID.

`-id` *`constraint_ID`*

Specifies the ID of the clock constraint to remove from the current scenario. You must specify either a clock or source name or its constraint ID.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes a clock source latency from the specified clock in the current scenario. If the specified source does not match a clock with a latency constraint in the current scenario, or if the specified ID does not refer to a clock with a latency constraint, this command fails.

Do not specify both the source and the ID.

Exceptions

You cannot use wildcards when specifying a clock name.

Examples

The following example removes the clock source latency from the specified clock.

```
remove_clock_latency -source my_clock
```

See Also

`set_clock_latency`

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_clock_uncertainty

Tcl command; removes a clock-to-clock uncertainty from the current timing scenario by specifying either its exact arguments or its ID.

```
remove_clock_uncertainty -from | -rise_from | -fall_from from_clock_list -to | -rise_to| -
fall_to to_clock_list -setup {value} -hold {value}
remove_clock_uncertainty -id constraint_ID
```

Arguments

`-from`

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the source clock list. Only one of the `-from`, `-rise_from`, or `-fall_from` arguments can be specified for the constraint to be valid.

`-rise_from`

Specifies that the clock-to-clock uncertainty applies only to rising edges of the source clock list. Only one of the `-from`, `-rise_from`, or `-fall_from` arguments can be specified for the constraint to be valid.

`-fall_from`

Specifies that the clock-to-clock uncertainty applies only to falling edges of the source clock list. Only one of the `-from`, `-rise_from`, or `-fall_from` arguments can be specified for the constraint to be valid.

`from_clock_list`

Specifies the list of clock names as the uncertainty source.

`-to`

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the destination clock list. Only one of the `-to`, `-rise_to` , or `-fall_to` arguments can be specified for the constraint to be valid.

`-rise_to`

Specifies that the clock-to-clock uncertainty applies only to rising edges of the destination clock list. Only one of the `-to`, `-rise_to` , or `-fall_to` arguments can be specified for the constraint to be valid.

`-fall_to`

Specifies that the clock-to-clock uncertainty applies only to falling edges of the destination clock list. Only one of the `-to`, `-rise_to` , or `-fall_to` arguments can be specified for the constraint to be valid.

`to_clock_list`

Specifies the list of clock names as the uncertainty destination.

`-setup`

Specifies that the uncertainty applies only to setup checks. If none or both `-setup` and `-hold` are present, the uncertainty applies to both setup and hold checks.

`-hold`

Specifies that the uncertainty applies only to hold checks. If none or both `-setup` and `-hold` are present, the uncertainty applies to both setup and hold checks.

`-id` *`constraint_ID`*

Specifies the ID of the clock constraint to remove from the current scenario. You must specify either the exact parameters to set the constraint or its constraint ID.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes a clock-to-clock uncertainty from the specified clock in the current scenario. If the specified arguments do not match clocks with an uncertainty constraint in the current scenario, or if the specified ID does not refer to a clock-to-clock uncertainty constraint, this command fails.

Do not specify both the exact arguments and the ID.

Examples

```
remove_clock_uncertainty -from Clk1 -to Clk2
remove_clock_uncertainty -from Clk1 -fall_to { Clk2 Clk3 } -setup
remove_clock_uncertainty 4.3 -fall_from { Clk1 Clk2 } -rise_to *
remove_clock_uncertainty 0.1 -rise_from [ get_clocks { Clk1 Clk2 } ] -fall_to { Clk3 Clk4 } -setup
remove_clock_uncertainty 5 -rise_from Clk1 -to [ get_clocks {*} ]
remove_clock_uncertainty -id $clockId
```

See Also

`remove_clock`

`remove_generated_clock`

`set_clock_uncertainty`

Designer Tcl Command Reference

remove_disable_timing

Tcl command; removes a disable timing constraint by specifying its arguments, or its ID. If the arguments do not match a disable timing constraint, or if the ID does not refer to a disable timing constraint, the command fails.

```
remove_disable_timing -from value -to value name -id name
```

Arguments

`-from` *`from_port`*

Specifies the starting port. The –from and –to arguments must either both be present or both omitted for the constraint to be valid.

`-to` *`to_port`*

Specifies the ending port. The –from and –to arguments must either both be present or both omitted for the constraint to be valid.

`name`

Specifies the cell name where the disable timing constraint will be removed. It is an error to supply both a cell name and a constraint ID, as they are mutually exclusive. No wildcards are allowed when specifying a clock name, either alone or in an accessor command1.

`-id` *`name`*

Specifies the constraint name where the disable timing constraint will be removed. It is an error to supply both a cell name and a constraint ID, as they are mutually exclusive. No wildcards are allowed when specifying a clock name, either alone or in an accessor command1.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
remove_disable_timing -from port1 -to port2 -id new_constraint
```

Designer Tcl Command Reference

remove_false_path

Tcl command; removes a false path from the current timing scenario by specifying either its exact arguments or its ID.

```
remove_false_path [-from from_list] [-to to_list] [-through through_list] [-id constraint_ID]
remove_false_path -id constraint_ID
```

Arguments

`-from` *`from_list`*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through` *`through_list`*

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

`-to` *`to_list`*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-id` *`constraint_ID`*

Specifies the ID of the false path constraint to remove from the current scenario. You must specify either the exact false path to remove or the constraint ID that refers to the false path constraint to remove.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes a false path from the specified clock in the current scenario. If the arguments do not match a false path constraint in the current scenario, or if the specified ID does not refer to a false path constraint, this command fails.

Do not specify both the false path arguments and the constraint ID.

Exceptions

You cannot use wildcards when specifying a clock name, either alone or in an Accessor command such as get_pins or get_ports.

Examples

The following example specifies all false paths to remove:

```
remove_false_path -through U0/U1:Y
```

The following example removes the false path constraint using its id:

```
set fpId [set_false_path –from [get_clocks c*] –through {topx/reg/*} –to [get_ports out15] ]
remove_false_path –id $fpId
```

See Also

set_false_path

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_generated_clock

Tcl command; removes the specified generated clock constraint from the current scenario.

```
remove_generated_clock {-name clock_name | -id constraint_ID }
```

Arguments

`-name` *`clock_name`*

Specifies the name of the generated clock constraint to remove from the current scenario. You must specify either a clock name or an ID.

`-id` *`constraint_ID`*

Specifies the ID of the generated clock constraint to remove from the current scenario. You must specify either an ID or a clock name that exists in the current scenario.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes the specified generated clock constraint from the current scenario. If the specified name does not match a generated clock constraint in the current scenario, or if the specified ID does not refer to a generated clock constraint, this command fails.

Do not specify both the name and the ID.

Exceptions

You cannot use wildcards when specifying a generated clock name.

Examples

The following example removes the generated clock constraint named "my_user_clock":

```
remove_generated_clock -name my_user_clock
```

See Also

create_generated_clock

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_input_delay

Tcl command; removes an input delay a clock on a port by specifying both the clocks and port names or the ID of the input_delay constraint to remove.

```
remove_input_delay -clock clock_name port_pin_list
remove_input_delay -id constraint_ID
```

Arguments

`-clock` *`clock_name`*

Specifies the clock name to which the specified input delay value is assigned.

`port_pin_list`

Specifies the port names to which the specified input delay value is assigned.

`-id` *`constraint_ID`*

Specifies the ID of the clock with the input_delay value to remove from the current scenario. You must specify either both a clock name and list of port names or the input_delay constraint ID .

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes an input delay from the specified clocks and port in the current scenario. If the clocks and port names do not match an input delay constraint in the current scenario, or if the specified ID does not refer to an input delay constraint, this command fails.

Do not specify both the clock and port names and the constraint ID.

Exceptions

You cannot use wildcards when specifying a clock name, either alone or in an accessor command.

Examples

The following example removes the input delay from CLK1 on port data1:

```
remove_input_delay -clock [get_clocks CLK1] [get_ports data1]
```

See Also

set_input_delay

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_max_delay

Tcl command; removes a maximum delay constraint from the current timing scenario by specifying either its exact arguments or its ID.

```
remove_max_delay [-from from_list] [-to to_list] [-through through_list]
remove_max_delay -id constraint_ID
```

Arguments

`-from from_list`

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through through_list`

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

`-to to_list`

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-id constraint_ID`

Specifies the ID of the maximum delay constraint to remove from the current scenario. You must specify either the exact maximum delay arguments to remove or the constraint ID that refers to the maximum delay constraint to remove.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes a maximum delay value from the specified clock in the current scenario. If the arguments do not match a maximum delay constraint in the current scenario, or if the specified ID does not refer to a maximum delay constraint, this command fails.

Do not specify both the maximum delay arguments and the constraint ID.

Exceptions

You cannot use wildcards when specifying a clock name, either alone or in an Accessor command.

Examples

The following example specifies a range of maximum delay constraints to remove:

```
remove_max_delay -through U0/U1:Y
```

See Also

set_max_delay

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_min_delay

Tcl command; removes a minimum delay constraint in the current timing scenario by specifying either its exact arguments or its ID.

```
remove_min_delay [-from from_list] [-to to_list] [-through through_list]
remove_min_delay -id constraint_ID
```

Arguments

`-from` *`from_list`*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through` *`through_list`*

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

`-to`*`to_list`*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-id` *`constraint_ID`*

Specifies the ID of the minimum delay constraint to remove from the current scenario. You must specify either the exact minimum delay arguments to remove or the constraint ID that refers to the minimum delay constraint to remove.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes a minimum delay value from the specified clock in the current scenario. If the arguments do not match a minimum delay constraint in the current scenario, or if the specified ID does not refer to a minimum delay constraint, this command fails.

Do not specify both the minimum delay arguments and the constraint ID.

Exceptions

You cannot use wildcards when specifying a clock name, either alone or in an accessor command.

Examples

The following example specifies a range of minimum delay constraints to remove:

```
remove_min_delay -through U0/U1:Y
```

See Also

`set_min_delay`

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_multicycle_path

Tcl command; removes a multicycle path constraint in the current timing scenario by specifying either its exact arguments or its ID.

```
remove_multicycle_path [-from from_list] [-to to_list] [-through through_list]
remove multicycle_path -id constraint_ID
```

Arguments

`-from` *`from_list`*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through` *`through_list`*

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

`-to`*`to_list`*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-id constraint_ID`

Specifies the ID of the multicycle path constraint to remove from the current scenario. You must specify either the exact multicycle path arguments to remove or the constraint ID that refers to the multicycle path constraint to remove.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes a multicycle path from the specified clock in the current scenario. If the arguments do not match a multicycle path constraint in the current scenario, or if the specified ID does not refer to a multicycle path constraint, this command fails.

Do not specify both the multicycle path arguments and the constraint ID.

Exceptions

You cannot use wildcards when specifying a clock name, either alone or in an accessor command.

Examples

The following example removes all paths between reg1 and reg2 to 3 cycles for setup check.

```
remove_multicycle_path -from [get_pins {reg1}] –to [get_pins {reg2}]
```

See Also

set_multicycle_path

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_output_delay

Tcl command; removes an ouput delay by specifying both the clocks and port names or the ID of the output_delay constraint to remove.

```
remove_output_delay -clock clock_name port_pin_list
remove_output_delay -id constraint_ID
```

Arguments

`-clock clock_name`

Specifies the clock name to which the specified output delay value is assigned.

`port_pin_list`

Specifies the port names to which the specified output delay value is assigned.

`-id constraint_ID`

Specifies the ID of the clock with the output_delay value to remove from the current scenario. You must specify either both a clock name and list of port names or the output_delay constraint ID .

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes an output delay from the specified clocks and port in the current scenario. If the clocks and port names do not match an output delay constraint in the current scenario, or if the specified ID does not refer to an output delay constraint, this command fails.

Do not specify both the clock and port names and the constraint ID.

Exceptions

You cannot use wildcards when specifying a clock name, either alone or in an accessor command.

Examples

The following example removes the output delay from CLK1 on port out1:

```
remove_output_delay -clock [get_clocks CLK1] [get_ports out1]
```

See Also

set_output_delay

Tcl Command Documentation Conventions

Designer Tcl Command Reference

rename_scenario

Tcl command; renames the specified timing scenario with the new name provided. You must provide a unique new name (that is, it cannot already be used by another timing scenario).

```
rename_scenario oldname -new newname
```

Arguments

oldname

Specifies the current name of the timing scenario.

-new *newname*

Specifies the new name to give to the timing scenario.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

This command changes the name of the timing scenario in the list of scenarios.

Example

```
rename_scenario scenario_A -new scenario_B
```

See Also

create_scenario

delete_scenario

Tcl documentation conventions

Designer Tcl Command Reference

report

The report command provides you with frequently-used information in a convenient format.

You can generate several different types of reports using this command, including:

- report (Status)
- report (Timing) for SmartFusion2, SmartFusion, IGLOO, ProASIC3, Fusion families
- report (Timing violations) for SmartFusion2, SmartFusion, IGLOO, ProASIC3, Fusion families
- report (Pin)
- report (Flip-flop)
- report (I/O Bank)
- report (Global Usage)
- report (Power)

set_clock_latency

Tcl command; defines the delay between an external clock source and the definition pin of a clock within SmartTime.

```
set_clock_latency -source [-rise][-fall][-early][-late] delay clock
```

Arguments

`-source`

Specifies the source latency on a clock pin, potentially only on certain edges of the clock.

`-rise`

Specifies the edge for which this constraint will apply. If neither or both rise are passed, the same latency is applied to both edges.

`-fall`

Specifies the edge for which this constraint will apply. If neither or both rise are passed, the same latency is applied to both edges.

`-invert`

Specifies that the generated clock waveform is inverted with respect to the reference clock.

`-late`

Optional. Specifies that the latency is late bound on the latency. The appropriate bound is used to provide the most pessimistic timing scenario. However, if the value of "-late" is less than the value of "-early", optimistic timing takes place which could result in incorrect analysis. If neither or both "-early" and "-late" are provided, the same latency is used for both bounds, which results in the latency having no effect for single clock domain setup and hold checks.

`-early`

Optional. Specifies that the latency is early bound on the latency. The appropriate bound is used to provide the most pessimistic timing scenario. However, if the value of "-late" is less than the value of "-early", optimistic timing takes place which could result in incorrect analysis. If neither or both "-early" and "-late" are provided, the same latency is used for both bounds, which results in the latency having no effect for single clock domain setup and hold checks.

`delay`

Specifies the latency value for the constraint.

`clock`

Specifies the clock to which the constraint is applied. This clock must be constrained.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Clock source latency defines the delay between an external clock source and the definition pin of a clock within SmartTime. It behaves much like an input delay constraint. You can specify both an "early" delay and a"late" delay for this latency, providing an uncertainty which SmartTime propagates through its

calculations. Rising and falling edges of the same clock can have different latencies. If only one value is provided for the clock source latency, it is taken as the exact latency value, for both rising and falling edges.

Examples

The following example sets an early clock source latency of 0.4 on the rising edge of main_clock. It also sets a clock source latency of 1.2, for both the early and late values of the falling edge of main_clock. The late value for the clock source latency for the falling edge of main_clock remains undefined.

```
set_clock_latency –source –rise –early 0.4 { main_clock  }
set_clock_latency –source –fall    1.2 { main_clock }
```

See Also

create_clock
create_generated_clock
Tcl Command Documentation Conventions
Designer Tcl Command Reference

set_clock_to_output

SDC command; defines the timing budget available inside the FPGA for an output relative to a clock.

```
set_clock_to_output delay_value -clock clock_ref [–max] [–min] [–clock_fall] output_list
```

Arguments

delay_value

Specifies the clock to output delay in nanoseconds. This time represents the amount of time available inside the FPGA between the active clock edge and the data change at the output port.

-clock *clock_ref*

Specifies the reference clock to which the specified clock to output is related. This is a mandatory argument.

-max

Specifies that *delay_value* refers to the maximum clock to output at the specified output. If you do not specify –max or –min options, the tool assumes maximum and minimum clock to output delays to be equal.

-min

Specifies that *delay_value* refers to the minimum clock to output at the specified output. If you do not specify –max or –min options, the tool assumes maximum and minimum clock to output delays to be equal.

-clock_fall

Specifies that the delay is relative to the falling edge of the reference clock. The default is the rising edge.

output_list

Provides a list of output ports in the current design to which delay_value is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

set_clock_uncertainty

Tcl command; specifies a clock-to-clock uncertainty between two clocks (from and to) and returns the ID of the created constraint if the command succeeded.

```
set_clock_uncertainty uncertainty -from | -rise_from | -fall_from from_clock_list -to | -
rise_to | -fall_to to_clock_list -setup {value} -hold {value}
```

Arguments

uncertainty

Specifies the time in nanoseconds that represents the amount of variation between two clock edges.

`-from`

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the source clock list. Only one of the `-from`, `-rise_from`, or `-fall_from` arguments can be specified for the constraint to be valid.

`-rise_from`

Specifies that the clock-to-clock uncertainty applies only to rising edges of the source clock list. Only one of the `-from`, `-rise_from`, or `-fall_from` arguments can be specified for the constraint to be valid.

`-fall_from`

Specifies that the clock-to-clock uncertainty applies only to falling edges of the source clock list. Only one of the `-from`, `-rise_from`, or `-fall_from` arguments can be specified for the constraint to be valid.

from_clock_list

Specifies the list of clock names as the uncertainty source.

`-to`

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the destination clock list. Only one of the `-to`, `-rise_to` , or `-fall_to` arguments can be specified for the constraint to be valid.

`-rise_to`

Specifies that the clock-to-clock uncertainty applies only to rising edges of the destination clock list. Only one of the `-to`, `-rise_to` , or `-fall_to` arguments can be specified for the constraint to be valid.

`-fall_to`

Specifies that the clock-to-clock uncertainty applies only to falling edges of the destination clock list. Only one of the `-to`, `-rise_to` , or `-fall_to` arguments can be specified for the constraint to be valid.

to_clock_list

Specifies the list of clock names as the uncertainty destination.

`-setup`

Specifies that the uncertainty applies only to setup checks. If none or both `-setup` and `-hold` are present, the uncertainty applies to both setup and hold checks.

`-hold`

Specifies that the uncertainty applies only to hold checks. If none or both `-setup` and `-hold` are present, the uncertainty applies to both setup and hold checks.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

The set_clock_uncertainty command sets the timing uncertainty between two clock waveforms or maximum clock skew. Timing between clocks have no uncertainty unless you specify it.

Examples

```
set_clock_uncertainty 10 -from Clk1 -to Clk2
set_clock_uncertainty 0 -from Clk1 -fall_to { Clk2 Clk3 } -setup
set_clock_uncertainty 4.3 -fall_from { Clk1 Clk2 } -rise_to *
set_clock_uncertainty 0.1 -rise_from [ get_clocks { Clk1 Clk2 } ] -fall_to { Clk3 Clk4 }
-setup
set_clock_uncertainty 5 -rise_from Clk1 -to [ get_clocks {*} ]
```

See Also

create_clock
create_generated_clock
remove_clock_uncertainty
Designer Tcl Command Reference

set_current_scenario

Tcl command; specifies the timing scenario for the Timing Analyzer to use. All commands that follow this command will apply to the specified timing scenario.

```
set_current_scenario name
```

Arguments

name

Specifies the name of the timing scenario to which to apply all commands from this point on.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

A timing scenario is a set of timing constraints used with a design. If the specified scenario is already the current one, this command has no effect.

After setting the current scenario, constraints can be listed, added, or removed, the checker can be invoked on the set of constraints, and so on.

This command uses the specified timing scenario to compute timing analysis.

Example

```
set_current_scenario scenario_A
```

See Also

get_current_scenario
Tcl Command Documentation Conventions
Designer Tcl Command Reference

set_disable_timing

Tcl command; disables timing arcs within a cell and returns the ID of the created constraint if the command succeeded.

```
set_disable_timing -from value -to value name
```

Arguments

-from *from_port*

Specifies the starting port. The –from and –to arguments must either both be present or both omitted for the constraint to be valid.

-to *to_port*

Specifies the ending port. The –from and –to arguments must either both be present or both omitted for the constraint to be valid.

name

Specifies the cell name where the timing arcs will be disabled.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
set_disable_timing -from A -to Y a2
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

set_external_check

SDC command; defines the external setup and hold delays for an input relative to a clock.

```
set_external_check delay_value -clock clock_ref [–setup] [–hold] [–clock_fall] input_list
```

Arguments

delay_value

Specifies the external setup or external hold delay in nanoseconds. This time represents the amount of time available inside the FPGA for the specified input after a clock edge.

`-clock` *clock_ref*

Specifies the reference clock to which the specified external check is related. This is a mandatory argument.

`-setup`

Specifies that *delay_value* refers to the setup check at the specified input. This is a mandatory argument if –hold is not used. You must specify either the -setup or -hold option.

`-clock_fall`

Specifies that the delay is relative to the falling edge of the reference clock. The default is the rising edge.

input_list

Provides a list of input ports in the current design to which *delay_value* is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

set_false_path

Tcl command; identifies paths that are considered false and excluded from the timing analysis in the current timing scenario.

```
set_false_path [-from from_list] [-through through_list] [-to to_list]
```

Arguments

`-from` *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through` *through_list*

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

`-to` *to_list*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

The `set_false_path` command identifies specific timing paths as being false. The false timing paths are paths that do not propagate logic level changes. This constraint removes timing requirements on these false paths so that they are not considered during the timing analysis. The path starting points are the input ports or register clock pins, and the path ending points are the register data pins or output ports. This constraint disables setup and hold checking for the specified paths.

The false path information always takes precedence over multiple cycle path information and overrides maximum delay constraints. If more than one object is specified within one -through option, the path can pass through any objects.

You must specify at least one of the `–from`, `-to`, or `–through` arguments for this constraint to be valid.

Examples

The following example specifies all paths from clock pins of the registers in clock domain clk1 to data pins of a specific register in clock domain clk2 as false paths:

```
set_false_path –from [get_clocks {clk1}] –to reg_2:D
```

The following example specifies all paths through the pin U0/U1:Y to be false:

```
set_false_path -through U0/U1:Y
```

See Also

Tcl Command Documentation Conventions

Designer Tcl Command Reference

set_input_delay

Tcl command; creates an input delay on a port list by defining the arrival time of an input relative to a clock in the current scenario.

```
set_input_delay delay_value -clock clock_ref [–max] [–min] [–clock_fall] input_list
```

Arguments

delay_value

Specifies the arrival time in nanoseconds that represents the amount of time for which the signal is available at the specified input after a clock edge.

`-clock` *clock_ref*

Specifies the clock reference to which the specified input delay is related. This is a mandatory argument. If you do not specify -max or -min options, the tool assumes the maximum and minimum input delays to be equal.

`-max`

Specifies that delay_value refers to the longest path arriving at the specified input. If you do not specify -max or -min options, the tool assumes maximum and minimum input delays to be equal.

`-min`

Specifies that delay_value refers to the shortest path arriving at the specified input. If you do not specify -max or -min options, the tool assumes maximum and minimum input delays to be equal.

`-clock_fall`

Specifies that the delay is relative to the falling edge of the clock reference. The default is the rising edge.

input_list

Provides a list of input ports in the current design to which delay_value is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.and IGLOOe, except ProASIC3 nano and ProASIC3L

Description

The set_input_delay command sets input path delays on input ports relative to a clock edge. This usually represents a combinational path delay from the clock pin of a register external to the current design. For in/out (bidirectional) ports, you can specify the path delays for both input and output modes. The tool adds input delay to path delay for paths starting at primary inputs.

A clock is a singleton that represents the name of a defined clock constraint. This can be:

- a single port name used as source for a clock constraint
- a single pin name used as source for a clock constraint; for instance reg1:CLK. This name can be hierarchical (for instance toplevel/block1/reg2:CLK)
- an object accessor that will refer to one clock: [get_clocks {clk}]

Examples

The following example sets an input delay of 1.2ns for port data1 relative to the rising edge of CLK1:

```
set_input_delay 1.2 -clock [get_clocks CLK1] [get_ports data1]
```

The following example sets a different maximum and minimum input delay for port IN1 relative to the falling edge of CLK2:

```
set_input_delay 1.0 -clock_fall -clock CLK2 –min {IN1}
set_input_delay 1.4 -clock_fall -clock CLK2 –max {IN1}
```

See Also

set_output_delay

Tcl Command Documentation Conventions

Designer Tcl Command Reference

set_max_delay

Tcl command; specifies the maximum delay for the timing paths in the current scenario.

```
set_max_delay delay_value [-from from_list] [-to to_list] [-through through_list]
```

Arguments

delay_value

Specifies a floating point number in nanoseconds that represents the required maximum delay value for specified paths.

- If the path starting point is on a sequential device, the tool includes clock skew in the computed delay.
- If the path starting point has an input delay specified, the tool adds that delay value to the path delay.

- If the path ending point is on a sequential device, the tool includes clock skew and library setup time in the computed delay.
- If the ending point has an output delay specified, the tool adds that delay to the path delay.

`-from from_list`

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-to to_list`

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-through through_list`

Specifies a list of pins, ports, cells, or nets through which the timing paths must pass.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

This command specifies the required maximum delay for timing paths in the current design. The path length for any startpoint in from_list to any endpoint in to_list must be less than delay_value.

The timing engine automatically derives the individual maximum delay targets from clock waveforms and port input or output delays.

The maximum delay constraint is a timing exception. This constraint overrides the default single cycle timing relationship for one or more timing paths. This constraint also overrides a multicycle path constraint.

You must specify at least one of the `–from`, `-to`, or `–through` arguments for this constraint to be valid.

Examples

The following example sets a maximum delay by constraining all paths from ff1a:CLK or ff1b:CLK to ff2e:D with a delay less than 5 ns:

```
set_max_delay 5 -from {ff1a:CLK ff1b:CLK} -to {ff2e:D}
```

The following example sets a maximum delay by constraining all paths to output ports whose names start by “out” with a delay less than 3.8 ns:

```
set_max_delay 3.8 -to [get_ports out*]
```

See Also

set_min_delay
remove_max_delay
Tcl Command Documentation Conventions
Designer Tcl Command Reference

set_min_delay

Tcl command; specifies the minimum delay for the timing paths in the current scenario.

```
set_min_delay delay_value [-from from_list] [-to to_list] [-through through_list]
```

Arguments

delay_value

Specifies a floating point number in nanoseconds that represents the required minimum delay value for specified paths.

- If the path starting point is on a sequential device, the tool includes clock skew in the computed delay.
- If the path starting point has an input delay specified, the tool adds that delay value to the path delay.
- If the path ending point is on a sequential device, the tool includes clock skew and library setup time in the computed delay.
- If the ending point has an output delay specified, the tool adds that delay to the path delay.

`-from` *`from_list`*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-to` *`to_list`*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-through` *`through_list`*

Specifies a list of pins, ports, cells, or nets through which the timing paths must pass.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

This command specifies the required minimum delay for timing paths in the current design. The path length for any startpoint in from_list to any endpoint in to_list must be less than delay_value.

The timing engine automatically derives the individual minimum delay targets from clock waveforms and port input or output delays.

The minimum delay constraint is a timing exception. This constraint overrides the default single cycle timing relationship for one or more timing paths. This constraint also overrides a multicycle path constraint.

You must specify at least one of the `–from`, `-to`, or `–through` arguments for this constraint to be valid.

Examples

The following example sets a minimum delay by constraining all paths from ff1a:CLK or ff1b:CLK to ff2e:D with a delay less than 5 ns:

```
set_min_delay 5 -from {ff1a:CLK ff1b:CLK} -to {ff2e:D}
```

The following example sets a minimum delay by constraining all paths to output ports whose names start by “out” with a delay less than 3.8 ns:

```
set_min_delay 3.8 -to [get_ports out*]
```

See Also

`set_max_delay`

`remove_min_delay`

Tcl Command Documentation Conventions

Designer Tcl Command Reference

set_multicycle_path

Tcl command; defines a path that takes multiple clock cycles in the current scenario.

```
set_multicycle_path ncycles [-setup] [-hold] [-from from_list[–through through_list[-to
to_list
```

Arguments

ncycles

Specifies an integer value that represents a number of cycles the data path must have for setup or hold check. The value is relative to the starting point or ending point clock, before data is required at the ending point.

`-setup`

Optional. Applies the cycle value for the setup check only. This option does not affect the hold check. The default hold check will be applied unless you have specified another set_multicycle_path command for the hold value.

`-hold`

Optional. Applies the cycle value for the hold check only. This option does not affect the setup check.

Note: If you do not specify "-setup" or "-hold", the cycle value is applied to the setup check and the default hold check is performed (*ncycles* -1).

`-from` *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through` *through_list*

Specifies a list of pins or ports through which the multiple cycle paths must pass.

`-to` *to_list*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Setting multiple cycle paths constraint overrides the single cycle timing relationships between sequential elements by specifying the number of cycles that the data path must have for setup or hold checks. If you change the multiplier, it affects both the setup and hold checks.

False path information always takes precedence over multiple cycle path information. A specific maximum delay constraint overrides a general multiple cycle path constraint.

If you specify more than one object within one -through option, the path passes through any of the objects.

You must specify at least one of the `–from`, `-to`, or `–through` arguments for this constraint to be valid.

Exceptions

Multiple priority management is not supported in Microsemi SoC designs. All multiple cycle path constraints are handled with the same priority.

Examples

The following example sets all paths between reg1 and reg2 to 3 cycles for setup check. Hold check is measured at the previous edge of the clock at reg2.

```
set_multicycle_path 3 -from [get_pins {reg1}] –to [get_pins {reg2}]
```

The following example specifies that four cycles are needed for setup check on all paths starting at the registers in the clock domain ck1. Hold check is further specified with two cycles instead of the three cycles that would have been applied otherwise.

```
set_multicycle_path 4 -setup -from [get_clocks {ck1}]
set_multicycle_path 2 -hold -from [get_clocks {ck1}]
```

See Also

remove_multicycle_path

Tcl Command Documentation Conventions

Designer Tcl Command Reference

set_output_delay

Tcl command; defines the output delay of an output relative to a clock in the current scenario.

```
set_output_delay delay_value -clock clock_ref [–max] [–min] [–clock_fall] output_list
```

Arguments

delay_value

Specifies the amount of time before a clock edge for which the signal is required. This represents a combinational path delay to a register outside the current design plus the library setup time (for maximum output delay) or hold time (for minimum output delay).

-clock *clock_ref*

Specifies the clock reference to which the specified output delay is related. This is a mandatory argument. If you do not specify -max or -min options, the tool assumes the maximum and minimum input delays to be equal.

-max

Specifies that delay_value refers to the longest path from the specified output. If you do not specify -max or -min options, the tool assumes the maximum and minimum output delays to be equal.

-min

Specifies that delay_value refers to the shortest path from the specified output. If you do not specify -max or -min options, the tool assumes the maximum and minimum output delays to be equal.

-clock_fall

Specifies that the delay is relative to the falling edge of the clock reference. The default is the rising edge.

output_list

Provides a list of output ports in the current design to which delay_value is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

The set_output_delay command sets output path delays on output ports relative to a clock edge. Output ports have no output delay unless you specify it. For in/out (bidirectional) ports, you can specify the path delays for both input and output modes. The tool adds output delay to path delay for paths ending at primary outputs.

Examples

The following example sets an output delay of 1.2ns for port OUT1 relative to the rising edge of CLK1:

```
set_output_delay 1.2 -clock [get_clocks CLK1] [get_ports OUT1]
```

The following example sets a different maximum and minimum output delay for port OUT1 relative to the falling edge of CLK2:

```
set_output_delay 1.0 -clock_fall -clock CLK2 –min {OUT1}
set_output_delay 1.4 -clock_fall -clock CLK2 –max {OUT1}
```

See Also

remove_output_delay
set_input_delay
Tcl Command Documentation Conventions
Designer Tcl Command Reference

st_commit

Tcl command; saves the changes made in SmartTime to the design (.adb) file

```
st_commit
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
st_commit
```

See Also

st_restore
Tcl documentation conventions
Designer Tcl Command Reference

st_create_set

Tcl command; creates a set of paths to be analyzed. Use the arguments to specify which paths to include. To create a set that is a subset of a clock domain, specify it with `-clock` and `-type`. To create a set that is a subset of an inter-clock domain set, specify it with `-source_clock` and `-sink_clock`. To create a set that is a subset (filter) of an existing named set, specify the set to be filtered with `-from_set`.

To create a set that is not derived from an existing set, you must provide both the `-source` *`pin_list`* and `-sink`*`pin_list`* derived. Otherwise, the `-source` and `-sink` arguments act as filters on the pins from the parent set. You must give each new set a unique name in the design.

```
st_create_set -name name
[-parent_set name ]
[-clockclock_id -type value ]
[-in_to_out]
[-source_clock clock_id -sink_clock clock_id]
[-source pin_list ] -sink pin_list ]
```

Arguments

`-name` *`name`*

Specifies a unique name for the newly create path set.

-parent_set *`name`*

Specifies the name of the set to filter.

-clock *clock_id*

Specifies that the set is to be a subset of the given clock domain. This argument is valid only if you also specify the -type argument.

-type *value*

Specifies the predefined set type on which to base the new path set. You can only use this argument with the -clock argument, not by itself.

Value	Description
reg_to_reg	Paths between registers in the design
async_to_reg	Paths from asynchronous pins to registers
reg_to_async	Paths from registers to asynchronous pins
external_recovery	The set of paths from inputs to asynchronous pins
external_removal	The set of paths from inputs to asynchronous pins
external_setup	Paths from input ports to registers
external_hold	Paths from input ports to registers
clock_to_out	Paths from registers to output ports

-in_to_out

Specifies that the set is based on the “Input to Output” set, which includes paths that start at input ports and end at output ports.

-source_clock *clock_id*

Specifies that the set will be a subset of an inter-clock domain set with the given source clock.

You can only use this option with the -sink_clock option, not by itself.

-sink_clock *clock_id*

Specifies that the set will be a subset of an inter-clock domain set with the given sink clock.

You can only use this option with the -source_clock option, not by itself.

-source *pin_list*

Specifies a filter on the source pins of the parent set. If you do not specify a parent set, this option filters all pins in the current design.

-sink *pin_list*

Specifies a filter on the sink pins of the parent set. If you do not specify a parent set, this option filters all pins in the current design.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
st_create_set –name { my_user_set } –source { C* } –sink { D* }
st_create_set –name { my_other_user_set } –from_set { my_user_set } –source { CL* }
st_create_set –name { adder } –clock { ALU_CLOCK } –type { REG_TO_REG }  -sink { ADDER*
}
st_create_set –name { another_set } –source_clock { EXTERN_CLOCK } –sink_clock {
MY_GEN_CLOCK }
```

```
st_create_set –name { some_p2p } –pin2pin –to { T* }
```

See Also

Designer Tcl Command Reference

Tcl documentation conventions

st_remove_set

st_edit_set

Tcl command; modifies the paths in a user set.

```
st_edit_set -name name
 [-source pin_list ] [-sink  pin_list ]
 [-rename_to name ]
```

Arguments

`-name name`

Specifies the name of the set to modify.

`-source pin_list`

If the set is a subset of another set, specifies a filter on the source pins from the parent set. Otherwise, this option specifies the source pins of the set.

`-sink pin_list`

If the set is a subset of another set, specifies a filter on the sink pins from the parent set. Otherwise, this option specifies the sink pins of the set.

`-rename_to name`

Specifies a new name for the set.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
st_edit_set –name { my_user_set} –rename_to { my_critical_pins }
st_edit_set –name { adder } –sink { ADD* }
```

See Also

Designer Tcl Command Reference

Tcl documentation conventions

st_create_set

st_remove_set

st_expand_path

Tcl command; displays expanded path information (path details) for paths. The paths to be expanded are identified by the parameters required to display these paths with st_list_paths. For example, to expand the first path listed with st_list_paths -clock {MYCLOCK} -type {register_to_register}, use the command st_expand_path -clock {MYCLOCK} -type {register_to_register}. Path details contain the pin name, type, net name, cell name, operation, delay, total delay, and edge as well as the arrival time, required time, and slack. These details are the same as details available in the SmartTime Expanded Path window.

```
st_expand_path [-set name]
[-clock clock_id -type value]
[-in_to_out]
```

```
[-source_clock clock_id -sink_clock clock_id]
[-source pin_list] [-sink pin_list]
[-analysis value]
[-index list_of_indices]
[-format value]
```

Arguments

-set *name*

Displays a list of paths from the named set. You can either use the -set option to specify a set name, or use both -clock and -type to specify a set. A list of valid set names includes "in_to_out", as well as any user set names.

-clock *clock_id*

Displays the set of paths belonging to the specified clock domain. You can either use this option along with -type to specify a set or use the -set option to specify the name of the set to display.

`-in_to_out`

Specifies that the paths should be from the set "Input to Output, which includes paths that start at input ports and end at output ports.

`-type` *value*

Specifies the type of paths in the clock domain to display in a list. You can only use this option with the -clock option, not by itself. You can either use this option along with -clock to specify a set or use the -set option to specify a set name.

Value	Description
reg_to_reg	Paths between registers in the design
async_to_reg	Paths from asynchronous pins to registers
reg_to_asyn	Paths from registers to asynchronous pins
external_recovery	The set of paths from inputs to asynchronous pins
external_removal	The set of paths from inputs to asynchronous pins
external_setup	Paths from input ports to registers
	Paths from input ports to registers
clock_to_out	Paths from registers to output ports

`-source_clock` *clock_id*

Displays a list of timing paths for an inter-clock domain set belonging to the source clock specified. You can only use this option with the -sink_clock option, not by itself.

`-sink_clock` *clock_id*

Displays a list of timing paths for an inter-clock domain set belonging to the sink clock specified. You can only use this option with the -source_clock option, not by itself.

`-source` *pin_list*

Specifies a filter on the source pins of the paths to be listed.

`-sink` *pin_list*

Specifies a filter on the sink pins of the paths to be listed.

`-analysis` *name*

Specifies the analysis type for the paths to be listed. The following table shows the acceptable values for this argument:

Value	Description
maxdelay	Maximum delay analysis
mindelay	Minimum delay analysis

`-index list_of_indices`

Specifies which paths to display. The index starts at 1 and defaults to 1. Only values lower than the max_paths option will be expanded.

`-format value`

Specifies the file format of the output. The following table shows the acceptable values for this argument:

Value	Description
text	ASCII text format
csv	Comma separated value fie format

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

Note: The following example returns a list of five paths:

```
st_expand_path –clock { myclock } –type {reg_to_reg }
st_expand_path –clock {myclock} –type {reg_to_reg} –index { 1 2 3 } –format text
```

See Also

Designer Tcl Command Reference

Tcl documentation conventions

st_list_paths

st_list_paths

Tcl command; displays the list of paths in the same tabular format shown in SmartTime.

```
st_list_paths [-set name ]
[-clock clock_id -type value ]
 [-in_to_out]
 [-source_clock clock_id -sink_clock  clock_id]
 [-source pin_list ] [-sink pin_list ]
 [-analysis value ]
 [-format value ]
```

Arguments

`-set name`

Displays a list of paths from the named set. You can either use the –set option to specify a set name, or use both –clock and –type to specify a set. A list of valid set names includes “in_to_out”, as well as any user set names.

-clock `clock_id`

Displays the set of paths belonging to the specified clock domain. You can either use this option along with -type to specify a set or use the -set option to specify the name of the set to display.

`-in_to_out`

Specifies that the paths should be from the set “Input to Output”, which includes paths that start at input ports and end at output ports.

`-type` *`value`*

Specifies the type of paths in the clock domain to display in a list. You can only use this option with the -clock option, not by itself. You can either use this option along with -clock to specify a set or use the -set option to specify a set name.

Value	Description
reg_to_reg	Paths between registers in the design
async_to_reg	Paths from asynchronous pins to registers
reg_to_asyn	Paths from registers to asynchronous pins
external_recovery	The set of paths from inputs to asynchronous pins
external_removal	The set of paths from inputs to asynchronous pins
external_setup	Paths from input ports to registers
	Paths from input ports to registers
clock_to_out	Paths from registers to output ports

`-source_clock` **`clock_id`**

Displays a list of timing paths for an inter-clock domain set belonging to the source clock specified. You can only use this option with the -sink_clock option, not by itself.

`-sink_clock` **`clock_id`**

Displays a list of timing paths for an inter-clock domain set belonging to the sink clock specified. You can only use this option with the -source_clock option, not by itself.

`-source` **`pin_list`**

Specifies a filter on the source pins of the paths to be listed.

`-sink` **`pin_list`**

Specifies a filter on the sink pins of the paths to be listed.

`-analysis` *`name`*

Specifies the analysis type for the paths to be listed. The following table shows the acceptable values for this argument:

Value	Description
maxdelay	Maximum delay analysis
mindelay	Minimum delay analysis

`-format` *`value`*

Specifies the file format of the output. The following table shows the acceptable values for this argument:

Value	Description
text	ASCII text format

Value	Description
csv	Comma separated value fie format

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
st_list_paths –set  { myset }
st_list_paths –analysis mindelay –clock { myclock } –type { reg_to_reg } –format csv
```

The list of paths can be written to a file with the following Tcl commands:

```
set outfile [ open “pathlisting.csv” w]
puts $outfile [ st_list_paths –format csv –set { myset} ]
close $outfile
```

See Also

Designer Tcl Command Reference
Tcl documentation conventions
st_expand_path

st_remove_all_constraints

Tcl command; removes all timing constraints from analysis

```
st_remove_all_constraints
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
st_remove_all_constraints
```

st_remove_set

Tcl command; deletes a user set from the design.

```
st_remove_set -name name
```

Arguments

`-name` *`name`*

Specifies the name of the set to delete.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
st_remove_set { clockset1 }
```

See Also

Designer Tcl Command Reference

Tcl documentation conventions

st_create_set

st_restore

Tcl command; restores constraints previously committed in SmartTime.

```
st_restore
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
st_restore
```

See Also

st_commit

Tcl documentation conventions

Designer Tcl Command Reference

st_set_options (SmartFusion, IGLOO, ProASIC3, Fusion only)

Tcl command; sets options for timing analysis. With no parameters given, it will display the current settings of the options. For SmartFusion, IGLOO, ProASIC3, Fusion families, these options also affect timing-driven place-and-route.

```
st_set_options [-max_opcond value ]
 [-min_opcond value]
 [-interclockdomain_analysis value]
 [-use_bibuf_loopbacks value]
 [-enable_recovery_removal_checks value]
 [-break_at_async value]
 [-filter_when_slack_below value]
 [-filter_when_slack_above value]
 [-remove_slack_filters]
 [-limit_max_paths value]
 [-expand_clock_network value]
 [-expand_parallel_paths value]
 [-analysis_scenario value]
 [-tdpr_scenario value]
 [-reset]
```

Arguments

-max_opcond *value*

Sets the operating condition to use for Maximum Delay Analysis. The following table shows the acceptable values for this argument:

Value	Description
worst	Use Worst Case conditions for Maximum Delay Analysis
typ	Use Typical conditions for Maximum Delay Analysis
best	Use Best Case conditions for Maximum Delay Analysis

-min_opcond *value*

Sets the operating condition to use for Minimum Delay Analysis. The following table shows the acceptable values for this argument:

Value	Description
best	Use Best Case conditions for Minimum Delay Analysis
typ	Use Typical conditions for Minimum Delay Analysis
worst	Use Worst Case conditions for Minimum Delay Analysis

-interclockdomain_analysis *value*

Enables or disables inter-clock domain analysis.

Value	Description
yes	Enables inter-clock domain analysis
no	Disables inter-clock domain analysis

-use_bibuf_loopbacks *value*

Enables or disables loopback in bibufs.

Value	Description
yes	Enables loopback in bibufs
no	Disables loopback in bibufs

-enable_recovery_removal_checks *value*

Enables or disables recovery and removal checks.

Value	Description
yes	Enables recovery and removal checks
no	Disables recovery and removal checks

-break_at_async *value*

Enables or disables breaking paths at asynchronous ports.

Value	Description
yes	Enables breaking paths at asynchronous ports
no	Disables breaking paths at asynchronous ports

-filter_when_slack_below *value*

Do not show paths with slack below x.

-filter_when_slack_above *value*

Do not show paths with slack above y.

-remove_slack_filters

Remove all existing slack filters.

-limit_max_paths *value*

Limit path reporting commands to a maximum of <n> paths, where n is a value of 0 or higher.

-expand_clock_network *value*

Enables or disables expanded clock network information in expanded paths.

Value	Description
yes	Enables expanded clock network information in paths
no	Disables expanded clock network information in paths

-expand_parallel_paths *value*

Expand a maximum of <n> parallel paths, where n is a value of 0 or higher. If n is 0 or 1, only one path will be expanded when viewing expanded paths.

-analysis_scenario *value*

Set the timing constraints scenario to be used for both maximum delay and minimum delay analysis. The argument must be a valid scenario name.

Note: This option does not affect the timing scenario used for TDPR.

-tdpr_scenario *value*

Set the timing constraints scenario to be used by the place and route engine. The argument must be a valid scenario name.

Note: This option does not affect the timing scenario used for analysis.

-reset

Reset all options to their default values, except for scenarios used for analysis and TDPR that remain unchanged.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
st_set_options –max_opcond worst
-min_opcond best
-interclockdomain_analysis true
-enable_removal_recovery_checks true
st_set_options –limit_max_paths 50 –remove_slack_filters
–filter_when_slack_above 3
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

timer_get_clock_actuals

Tcl command; displays the actual clock frequency in the Log window, when the timing analysis tool is initiated.

```
timer_get_clock_actuals -clock clock_name
```

Arguments

-clock *clock_name*

Specifies the name of the clock with the frequency (or period) to display.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

This example displays the actual clock frequency of clock clk1 in the Log window:

```
timer_get_clock_actuals -clock clk1
```

See Also

timer_get_clock_constraints

Tcl documentation conventions

Designer Tcl Command Reference

timer_get_clock_constraints

Tcl command; returns the constraints (period, frequency, and duty cycle) on the specified clock.

```
timer_get_clock_constraints -clock clock_name
```

Arguments

-clock *clock_name*

Specifies the name of the clock with the constraint to display.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

The following example displays the clock constraints on the clock clk in the Log window:

```
timer_get_clock_constraints -clock clk
```

See Also

timer_get_clock_actuals

Tcl documentation conventions

Designer Tcl Command Reference

timer_get_maxdelay

Tcl command; displays the maximum delay constraint between two pins in a path in the Log window.

```
timer_get_maxdelay -from source_pin -to destination_pin
```

Arguments

`-from source_pin`

Specifies the name of the source pin in the path.

`-to destination_pin`

Specifies the name of the destination pin in the path.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

The following example displays the maximum delay constraint from the pin clk166 to the pin reg_q_a_9_/U0:CLK in the Log window:

```
timer_get_maxdelay -from {clk166} -to {reg_q_a_9_/U0:CLK}
```

See Also

`timer_set_maxdelay`

Tcl documentation conventions

Designer Tcl Command Reference

timer_get_path

Tcl command; displays the path between the specified pins in the Log window.

```
timer_get_path -from source_pin -to destination_pin
[-exp value]\
[-sort value]\
[-order value]\
[-case value]\
[-maxpath maximum_paths]\
[-maxexpath maximum_paths_to_expand]\
[-mindelay minimum_delay]\
[-maxdelay maximum_delay]\
[-breakatclk value]\
[-breakatclr value]
```

Arguments

`-from source_pin`

Specifies the name of the source pin for the path.

`-to destination_pin`

Specifies the name of the destination pin for the path.

`-exp value`

Specifies whether to expand the path. The following table shows the acceptable values for this argument:

Value	Description
yes	Expands the path

Value	Description
no	Does not expand the path

`-sort` *value*

Specifies whether to sort the path by either the actual delay or slack value. The following table shows the acceptable values for this argument:

Value	Description
actual	Sorts the path by the actual delay value
slack	Sorts the path by the slack value

`-order` *value*

Specifies whether the list is based on maximum or minimum delay analysis. The following table shows the acceptable values for this argument:

Value	Description
long	The paths are listed based on the maximum delay analysis
short	The paths are listed based on the minimum delay analysis

`-case` *value*

Specifies whether the report will include the worst, typical, or best case timing numbers. The following table shows the acceptable values for this argument:

Value	Description
worst	Includes worst case timing numbers
typ	Includes typical case timing numbers
best	Includes best case timing numbers

`-maxpath` *maximum_paths*

Specifies the maximum number of paths to display.

`-maxexpath` *maximum_paths_to_expand*

Specifies the maximum number of paths to expand.

`-mindelay` *minimum_delay*

Specifies the path delay in the minimum delay analysis mode.

`-maxdelay` *maximum_delay*

Specifies the path delay in the maximum delay analysis mode.

`-breakatclk` *value*

Specifies whether to break the paths at the register clock pins. The following table shows the acceptable values for this argument:

Value	Description
yes	Breaks the paths at the register clock pins

Value	Description
no	Does not break the paths at the register clock pins

`-breakatclr` *value*

Specifies whether to break the paths at the register clear pins. The following table shows the acceptable values for this argument:

Value	Description
yes	Breaks the paths at the register clear pins
no	Does not break the paths at the register clear pins

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

The following example returns the paths from input port headdr_dat<31> to the input pin of register u0_headdr_data1_reg/data_out_t_31 under typical conditions.

```
timer_get_path -from "headdr_dat<31>" \
-to  "u0_headdr_data1_reg/data_out_t_31/U0:D" \
-case typ  \
-exp "yes" \
-maxpath "100" \
-maxexpath "10"
```

The following example returns the paths from the clock pin of register gearbox_inst/bits64_out_reg<55> to the output port pma_tx_data_64bit[55]

```
timer_get_path -from "gearbox_inst/bits64_out_reg<55>/U0:CLK" \
     -to {pma_tx_data_64bit[55]} \
     -exp "yes"
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

timer_get_path_constraints

Tcl command; displays the path constraints that were set as the maximum delay constraint in the timing analysis tool.

```
timer_get_path_constraints
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

This command lists the paths constrained by maximum delay values. The information is displayed in the Log window. If no maximum delay constraints are set, this command does not report anything.

Examples

Invoking timer_get_path_constraints displays the following paths and their delay constraints in the Log window:

```
max_delay -from [all_inputs] -to [all_outputs] = 12 ns
max_delay -from p_f_testwdata0 p_f_testwdata1 -to p_f_dacuwdata0 p_f_dacuwdata1
r_f_dacuwdata0 r_f_dacuwdata1 = 8 ns
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

timer_remove_all_constraints

Tcl command; removes all timing constraints in the current design.

```
timer_remove_all_constraints
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

The following example removes all of the constraints from the design and then commits the changes:

```
timer_remove_all_constraints
timer_commit
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

timer_remove_stop

Tcl command; removes the previously entered path stop constraint on the specified pin.

```
timer_remove_stop -pin pin_name
```

Arguments

`-pin pin_name`

Specifies the name of the pin from which to remove the path stop constraint.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

If you remove a path stop constraint using the Timer GUI, and then export a script using **File > Export** > **Script files**, the resulting script will contain `timer_remove_pass -pin pin_name` instead of `timer_remove_stop -pin pin_name`.

Exceptions

- For the SmartFusion2, SmartFusion, IGLOO, ProASIC3, Fusion families, best practice is to use the following flow:
    1. Open **SmartTime** > **Set False Path Constraint dialog box**.
    2. Look for the pin name in the **Through:** list (Note: You must not have any entry selected in the **From** or **To** lists).
    3. Delete this pin.

Examples

The following example removes the path stop constraint on the clear pin reg_q_a_0_:CLR:

```
timer_remove_stop -pin {reg_q_a_0_:CLR}
```

See Also

Tcl documentation conventions

Set False Path Constraint dialog box

Designer Tcl Command Reference

timer_restore

Tcl command; restores constraints previously committed in Timer.

```
timer_restore
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
timer_restore
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

Tcl Flow in the Libero SoC

Use the following commands to manage and build your project in the Libero SoC.

Design Flow in the Project Manager

The Tcl commands below outline the entire design flow. Once you create a project in the Project Manager you can use the commands below to complete every operation from synthesis to generating an HDL netlist. Click any command to go to the command definition.

run_synthesis [-logfile *name*]

run_simulation [-logfile *name*]

check_hdl -file *filename*

check_schematic -file *filename*

create_symbol [-module *module*]

export_io_constraints_from_adb -adb *filename* -output *outputfilename*

generate_ba_files -adb *filename*

generate_hdl_from_schematic [-module *modulename*]

generate_hdl_netlist [-netlist *filename*] [-run_drc *"TRUE | FALSE"*]

rollback_constraints_from_adb -adb *filename* -output *output_filename*

run_designer [-logfile *filename*] [-script "*script to append*"] [-append_commands "*commands to execute*"] [-adb "*new | open | default*"] [-compile "*TRUE | FALSE*"] [-layout "*TRUE | FALSE*"] [-export_ba "*TRUE | FALSE*"]

run_drc [-netlist *file*] [-gen_hdl *"TRUE | FALSE"*]

Manage Profiles in the Project Manager

add_profile -name *profilename* -type "*synthesis | simulation | stimulus | flashpro | physynth | coreconfig*" -tool *profiletool* -location *tool_location* [-args *tool_parameters*] [-batch "*TRUE | FALSE*"]

edit_profile -name *profilename* -type "*synthesis | simulation | stimulus | flashpro | physynth | coreconfig*" -tool *profiletool* -location *tool_location* [-args *tool_parameters*] [-batch "*TRUE | FALSE*"] [-new_name *name*]

export_profiles -file *name* [-export "*predefined | user | all*"]

remove_profile -name *profile_name*

select_profile -name *profile_name*

Linking Files

change_link_source -file *filename* -path *pathname*

create_links [-hdl_source *file*]* [-stimulus *file*]* [-sdc *file*]* [-pin *file*]* [-dcf *file*]* [-gcf *file*]* [-pdc *file*]* [-crt *file*]* [-vcd *file*]*

export_as_link -file *filename* -path *link_path*

unlink -file *file* [-local *local_filename*]

Set Simulation Options in the Project Manager

add_modelsim_path -lib *library_name* [-path *library_path*] [-remove " "]

Set Device in the Project Manager

set_device [-family *family*] [-die *die*] [-package *package*]

Miscellaneous Operations in the Project Manager

project_settings [-hdl "*VHDL | VERILOG*"] [-auto_update_modelsim_ini "*TRUE | FALSE*"] [-auto_update_viewdraw_ini "*TRUE | FALSE*"] [-block_mode "*TRUE | FALSE*"] [-

```
auto_generate_synth_hdl "TRUE | FALSE"] [-auto_run_drc "TRUE | FALSE"] [-
auto_generate_viewdraw_hdl "TRUE | FALSE"] [-auto_file_detection "TRUE | FALSE"]
refresh
set_option [-synth "TRUE | FALSE"] [-module "module_name"]
remove_core -name core_name
```

SmartFusion, IGLOO, ProASIC3, and Fusion – Project Manager

add_probe (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; adds a probe to an internal net in your design, using the original name from the optimized netlist in your design. Also, this command must be used in conjunction with the generate_probes command to generate a probed ADB file (see example below).

You must complete layout before you use this command.

```
add_probe –net <net_name> [-pin <pin_name>] [-port <port_name>] [-assign_to_used_pin <TRUE|FALSE>]
```

Arguments

`-net <net_name>`

Name of the net you want to probe. You cannot probe HARDWIRED, POWER, or INTRINSIC nets.

`-pin <pin_name>`

Name of the package pin at which you want to put the net to be probed. Argument is optional; if unspecified the net is routed to any free package pin.

`-port <port_name>`

Name of the port you are adding. Argument is optional; if unspecified the default value is PROBE_<n>.

`-assign_to_used_pin <TRUE|FALSE>`

Probes a net on an already used pin. The net on the existing pin will be disconnected. Argument is optional; if unspecified the net can be only routed on unused pin.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The example below adds a probe to the net Count8_0/INV_0_Y on pin 7 and uses the port name PROBE_1, then generates the probe ADB file named test1.adb.

Note that generate_probes is a separate Tcl command.

```
add_probe -net Count8_0/INV_0_Y -assign_to_used_pin {FALSE} -pin {7} -port {PROBE_1}
generate_probes –save test1.adb
```

See Also

`delete_probe`

`generate_probes`

Generating a Probed Design

Generate Probed Design - Add Probe(s) Dialog Box

Designer Tcl Command Reference

are_all_source_files_current (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; audits all source files and determines whether or not they are out of date / imported into the workspace. Returns '1' if all source files are current Returns '0' if all source files are not current This command ignores the Audit settings in your ADB file.

```
are_all_source_files_current
```

Arguments

None

Supported Family

See the Tcl Commands and Supported Families table for the list of families that support this command.

Exceptions

The command will return an error if arguments are passed.

Example

The following code will determine if your source files are current.

```
are_all_source_files_current
```

See Also

get_out_of_date_files
is_source_file_current
Designer Tcl Command Reference

backannotate (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; equivalent to executing the Back-Annotate command from the Tools menu. You can export an SDF file, after layout, along with the corresponding netlist in the VHDL or Verilog format. These files are useful in backannotated timing simulation.

Best practice is to export both SDF and the corresponding VHDL/Verilog files. This will avoid name conflicts in the simulation tool.

Designer must have completed layout before this command can be invoked, otherwise the command will fail.

```
backannotate -name file_name -format format_type -language language-dir directory_name [-
netlist] [-pin] [-use_emd]
```

Arguments

`-name file_name`

Use a valid file name with this option. You can attach the file extension .sdf to the File_Name, otherwise the tool will append .sdf for you.

`-format format_type`

Only SDF format is available for back annotation

`-language language`

The supported Language options are:

Value	Description

Value	Description
VHDL93	For VHDL-93 style naming in SDF
VERILOG	For Verilog style naming in SDF

`-dir` *`directory_name`*

Specify the directory in which all the files will be extracted.

`-netlist`

Forces a netlist to be written. The netlist will be either in Verilog or VHDL.

`-pin`

Designer exports the pin file with this option. The .pin file extension is appended to the design name to create the pin file.

`-use_emd`

Enables Export Enhanced Min Delays for Best Case option in your backannotated file.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

Example 1 uses default arguments and exports SDF file for back annotation:

```
backannotate
```

Example 2 uses some of the options for VHDL:

```
backannotate -dir \
 {..\my_design_dir} -name "fanouttest_ba.sdf" -format "SDF" –language \ "VHDL93" –netlist
```

Example 3 uses some of the options for Verilog:

```
backannotate -dir \
 {..\design} -name "fanouttest_ba.sdf" -format "SDF" -language "VERILOG" \
-netlist
```

Example 4 enables you to catch exceptions and respond based on the success of backannotate operation:

```
If  { [catch { backannotate -name "fanouttest_ba" -format "SDF" } ]} {
              Puts “Back annotation failed”
              # Handle Failure
} else {
              Puts “Back annotation successful”
              # Proceed with other operations
}
```

Example 5 enables Export Enhanced Min Delays for Best Case:

```
backannotate -dir \ {..\my_design_dir} -name "fanouttest_ba.sdf" -format "SDF"
–language \ "VHDL93" –netlist -use_emd
```

See Also

Tcl command documentation conventions

Designer Tcl Command Reference

close_design (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; closes the current design and brings Designer to a fresh state to work on a new design. This is equivalent to selecting the Close command from the File menu.

```
close_design
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
if  { [catch { close_design }] {
            Puts “Failed to close design”
            # Handle Failure
} else {
            puts “Design closed successfully”
            # Proceed with processing a new design
}
```

See Also

open_design
close_design
new_design
Designer Tcl Command Reference

configure_tool (SmartFusion, IGLOO, ProASIC3 and Fusion)

configure_tool is a general-purpose Tcl command to set the parameters for any tool called by Libero for the SmartFusion, IGLOO, ProASIC3 and Fusion families. The command requires the name of the tool and one or more parameters in the format *tool_parameter*:*value*. These parameters are separated and passed to the tool to set up its run.

```
configure_tool
-name {<tool_name>} # Each tool_name has its own set of parameters
-params {<parameter>:<value>} # List of parameters and values
tool_name ::= COMPILE | SYNTHESIZE | PLACEROUTE | PUBLISHBLOCK
```

Supported tool_names

The following table lists the supported tool_names.

tool_name	Parameter (-params)	Description
COMPILE	See the topic for parameter names and values.	See the topic for description.
SYNTHESIZE	See the topic for parameter names and values.	See the topic for description.
PLACEROUTE	See the topic for parameter names and values.	See the topic for description.
PUBLISHBLOCK	See the topic for parameter names and values.	See the topic for description.

See the SmartFusion2 and IGLOO2 Tcl for SoC document for the full list of parameters and values.

Example

```
configure_tool -name {SYNTHESIZE} -params {LANGUAGE_VHDL_2008:true}
configure_tool -name {COMPILE} \
  -params {MERGESDC:true}
configure_tool -name {PLACEROUTE} -params {TDPR:true}\
  –params {INCRPLACEANDROUTE:true}
```

For example, the command:

```
configure_tool -name {COMPILE} -params {MERGESDC:true}
```

sets the COMPILE command options MERGESDC to true to merge SDC file(s) with existing timing constraints.

There are alternative ways to write these commands to fit your coding style. The following three examples all do the same thing.

Method 1 - single line

```
configure_tool -name {COMPILE} -params {MERGESDC:true} -params\ {MERGEPDC:true}
```

Method 2 - one statement, multiple lines

```
configure_tool \
  -name {COMPILE} \
  -params {MERGEPDC:true}\
  -params {MERGESDC:true}
```

Method 3 - multiple statements

```
configure_tool -name {COMPILE} -params {MERGEPDC:true}
configure_tool -name {COMPILE} -params {MERGESDC:true}
configure_tool -name {COMPILE} \
  -params { DISPLAY_FANOUT_LIMIT:10}\
  -params {MERGE_SDC:true}
configure_tool -name {SYNTHESIZE} -params {LANGUAGE_VHDL_2008:true}
configure_tool -name {PLACEROUTE} -params {PDPR:false} -params \
  {TDPR:true} -{EFFORT_LEVEL:false} –params {INCRPLACEANDROUTE:false}
```

For example, the command:

```
configure_tool \
  -name {COMPILE} -params {DISPLAY_FANOUT_LIMIT:10} \
  -params {MERGE_SDC:true}
```

sets the COMPILE command options DISPLAY_FANOUT_LIMIT to 10 and MERGE_SDC to true.

There are alternative ways to write these commands to fit your coding style. The following three examples all do the same thing.

Method 1 - single line

```
configure_tool -name {COMPILE} -params {DISPLAY_FANOUT_LIMIT:10} -params {MERGE SDC:true}
```

Method 2 - one statement, multiple lines

```
configure_tool \
  -name {COMPILE} \
  -params {DISPLAY_FANOUT_LIMIT:10} \
  -params {MERGE_SDC:true}
```

Method 3 - multiple statements

```
configure_tool -name {COMPILE} -params {DISPLAY_FANOUT_LIMIT:10}
configure_tool -name {COMPILE} -params {MERGE_SDC:true}
```

See Also

Project Manager Tcl Command Reference

Tcl documentation conventions

.

delete_probe (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; deletes a probe on nets in a probed ADB file.

```
delete_probe –net <net_name>
```

Arguments

`-net <net_name>`

Name of the net you want to delete.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The example below deletes the probe on the net Count8_0/INV_0_Y.

```
delete_probe -net Count8_0/INV_0_Y
```

See Also

add_probe

Generating a Probed Design

Generate Probed Design - Add Probe(s) Dialog Box

Designer Tcl Command Reference

export_bitstream_file (SmartFusion, IGLOO, ProASIC3, and Fusion)

Configures the parameters for the bitstream to be exported from Libero.

```
 export_bitstream_file [-file_name file_name]
[-format STP | CHAIN_STP | DAT | SPI]
[-master_file 0 | 1]
[-master_file_components SECURITY | FABRIC | ENVM]
[-encrypted_uek1_file 1 | 0]
[-encrypted_uek1_file_components FABRIC | ENVM ]
[-encrypted_uek2_file 1 | 0]
[-encrypted_uek2_file_components "FABRIC | ENVM "]
[-trusted_facility_file 1 | 0]
[-trusted_facility_file_components FABRIC | ENVM ]
[-add_golden_image 1 | 0]
[-golden_image_address golden_image_address]
[-golden_image_design_version golden_image_design_version]
[-add_update_image 1 | 0]
[-update_image_address update_image_address]
[-update_image_design_version update_image_design_version]
[-serialization_stapl_type serialization_stapl_type]
[-serialization_target_solution serialization_target_solution]
```

Arguments

`-file_name` *`file_name`*

File name must start with design name. Bitstream files will be saved in Libero /export folder.

`-format` *`STP | CHAIN_STP | DAT | SPI`*

Specifies the programming file formats to be exported. Space or comma can be used as a delimiter.

`-master_file/encrypted_uek1_file/encrypted_uek2_file` *`1|0`*

Specifies the bitstream files to be exported. Depends on the selected security.

`-master_file_components` *`security fabric envm`*

Specifies the components in the design that will be saved to the programming file.

`-encrypted_uek1_file_components/encrypted_uek2_file_components` *`fabric | envm`*

Specifies the components of the design that will be saved to uek1/uek2 bitstreams.

SPI related commands:

`-add_golden_image`

To enable/disable golden SPI image in SPI directory.

`-golden_image_address`

Hexadecimal address for golden image.

`-golden_image_design_version`

Decimal value for golden image design version.

`-add_update_image`

To enable/disable update SPI image.

`-update_image_address`

Hexadecimal value for update image address.

`-update_image_design_version`

Decimal value for update image design version.

Serialization options:

`-serialization_stapl_type`

Serialization stapl file type either single or multiple.

`-serialization_target_solution`

Target programming hardware – Flashpro_3_4_5 or generic_STAPL_player.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Export a bitstream file:

```
export_bitstream_file\
  -file_name {iaptcl1}\
  -format {STP }\
  -master_file 1\
  -master_file_components {SECURITY FABRIC}\
  -encrypted_uek1_file 1\
  -encrypted_uek1_file_components {FABRIC }\
  -encrypted_uek2_file 0\
  -encrypted_uek2_file_components { }\
  -trusted_facility_file 0\
  -trusted_facility_file_components { }\
  -add_golden_image 1\
  -golden_image_address {3}\
  -golden_image_design_version {3}\
```

```
-add_update_image 1\
-update_image_address {3}\
-update_image_design_version {3}\
-serialization_stapl_type {SINGLE}\
-serialization_target_solution {FLASHPRO_3_4_5}
```

See Also

Project Manager Tcl Command Reference

export_io_constraints_from_adb

Tcl command; exports the I/O constraints from your project ADB file to an output file.

```
export_io_constraints_from_adb -adb filename -output outputfilename
```

Arguments

`-adb filename`

Specifies name of the ADB file from which you want to export your I/O constraints.

`-output filename`

Specifies the output filename for your exported I/O constraints.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following example exports the I/O constraint file ios.pdc from the project file designer1.adb:

```
export_io_constraints_from_adb -adb designer1.adb -output ios.pdc
```

See Also

Project Manager Tcl Command Reference

generate_ba_files

Tcl command; generates the back-annotate files for your design.

```
generate_ba_files -adb filename
```

Arguments

`-adb filename`

Specifies name of the ADB file from which you wish to generate the backannotate files.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following example generates backa-nnotate files from the file designer1.adb:

```
generate_ba_files -adb designer1.adb
```

See Also

Project Manager Tcl Command Reference

generate_hdl_netlist (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; generates the HDL netlist for your design and runs the design rule check.

```
generate_hdl_netlist [-netlist filename] [-run_drc value]
```

Arguments

`-netlist filename`

Specifies the filename of your netlist.

`-run_drc value`

Runs the design rule check. The following table shows the acceptable values for this argument:

Value	Description
TRUE	Runs the design rule check
FALSE	Generates your netlist without running the design rule check

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following example generates your netlist netlist2 and runs the design rule check:

```
generate_hdl_netlist [-netlist netlist2][-run_drc TRUE]
```

See Also

Project Manager Tcl Command Reference

generate_probes (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; executes the probing and creates a new ADB file. This command is used in conjunction with the add_probe Tcl command (see example below).

```
generate_probes –save <ADB_file_name>
```

Arguments

`-save <ADB_file_name>`

Name of the new ADB file with your probed nets.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The example below adds a probe to the net net2 on pin 4 and port prb2 with the add_probe command, and generates the new ADB file test1.adb.

```
add_probe –net net2 –pin 4 –port prb2
generate_probes –save test1.adb
```

See Also

add_probe

Generating a Probed Design

Generate Probed Design - Add Probe(s) Dialog Box

Designer Tcl Command Reference

get_defvar (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; provides access to the internal variables within Designer and returns its value. This command also prints the value of the Designer variable on the Log window.

```
get_defvar variable
```

Arguments

variable

The Designer internal variable.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Example 1: Prints the design name on the log window.

```
   get_defvar “DESIGN”
set variableToGet "DESIGN"
set valueOfVariable [get_defvar $variableToGet]
puts "The value is $valueOfVariable"
```

See Also

set_defvar

Designer Tcl Command Reference

get_design_filename (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; retrieves the full qualified path of the design file. The result will be an empty string if the design has not been saved to disk. This command is equivalent to the command “get_design_info DESIGN_PATH.” This command predates get_design_info and is supported for backward-compatibility.

```
get_design_filename
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Exceptions

- The command will return an error if a design is not loaded.
- The command will return an error if arguments are passed.

Example

```
if { [ is_design_loaded ] } {
  set design_location [ get_design_filename ]
  if {$design_location != "" } {
    puts “Design is at $design_location.”
  } else {
    puts “Design has not been saved to a file on disk.”
  }
} else {
  puts "No design is loaded."
}
```

See Also

get_design_info
is_design_loaded
is_design_modified
is_design_state_complete
Designer Tcl Command Reference

get_design_info (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; retrieves some basic details of your design. The result value of the command will be a string value.

```
get_design_info value
```

Arguments

value

Must be one of the valid string values summarized in the table below:

Value	Description
name	Design name. The result is set to the design name string.
family	Silicon family. The result is set to the family name.
design_path	Fully qualified path of the design file. The result is set to the location of the .adb file. If a design has not been saved to disk, the result will be an empty string. This command replaces the command get_design_filename.
design_folder	Directory (folder) portion of the design_path.
design_file	Filename portion of the design_path.

Value	Description
cwdir	Current working directory. The result is set to the location of the current working directory
die	Die name. The result is set to the name of the selected die for the design. If no die is selected, this is an empty string.
Package	Package. The result is set to the name of the selected package for the design. If no package is selected, this is an empty string.
Speed	Speed grade. The result is set to the speed grade for the design. If no speed grade is selected, this is an empty string.

Supported Family

See the Tcl Commands and Supported Families table for the list of families that support this command.

Exceptions

- Returns an error if a design is not loaded.
- Returns an error if more than one argument is passed.
- Returns an error if the argument is not one of the valid values.

Example

The following example uses get_design_info to display the various values to the screen.

```
if { [ is_design_loaded ] } {
  puts "Design is loaded."
  set bDesignLoaded 1
} else {
  puts "No design is loaded."
  set bDesignLoaded 0
}
if { $bDesignLoaded != 0 } {
  set var [ get_design_info NAME ]
  puts "  DESIGN NAME:\t$var"
  set var [ get_design_info FAMILY ]
  puts "  FAMILY:\t$var"
  set var [ get_design_info DESIGN_PATH ]
  puts "  DESIGN PATH:\t$var"
  set var [ get_design_info DESIGN_FILE ]
  puts "  DESIGN FILE:\t$var"
  set var [ get_design_info DESIGN_FOLDER ]
  puts "  DESIGN FOLDER:\t$var"
  set var [ get_design_info CWDIR ]
  puts "  WORKING DIRECTORY:  $var"
  set var [ get_design_info DIE ]
  puts "  DIE:\t$var"
  set var [ get_design_info PACKAGE ]
```

```
  puts "  PACKAGE:\t'$var'"
  set var [ get_design_info SPEED ]
  puts "  SPEED GRADE:\t$var"
  if { [ is_design_modified ] } {
    puts "The design is modified."
  } else {
    puts "The design is unchanged"
  }
}
puts "get_design.tcl done"
```

See Also

[get_design_filename]
[is_design_loaded]
[is_design_modified]
[is_design_state_complete]
Designer Tcl Command Reference

get_out_of_date_files (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; audits all files returns a list of filenames that are out of date; each filename is separated by a space. The command returns a string of file names that are out of date separated by a space i.e. file1 file2 ...

It returns empty string if all files are current.

This command ignores the Audit settings in your ADB file.

```
get_out_of_date_files
```

Arguments

None

Supported Family

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following code returns a list of filenames that are out of date.

```
get_out_of_date_files
```

See Also

[are_all_source_files_curent]
[is_source_file_current]
Designer Tcl Command Reference

ioadvisor_apply_suggestion (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; applies the suggestions for the selected attribute to the selected I/O(s).

```
ioadvisor_apply_suggestion -attribute {value} -io {value}
```

Arguments

`-attribute{value}`

This specifies the attribute for which the values will be applied. The following table shows the acceptable values for this argument:

Value	Description
outdrive	Applies suggested outdrive values
slew	Applies suggested slew values

`-io {value}`

This selects the I/Os for which the suggestion will be applied. To select multiple I/Os, use `-io {value}` for each I/O.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following code applies the suggested outdrive values for two I/Os.

```
ioadvisor_apply_suggestion -attribute{outdrive} -io{nPWM_out_pad} -io{PWM_out_pad}
```

See Also

ioadvisor_commit
ioadvisor_restore
ioadvisor_restore_initial_value
ioadvisor_set_outdrive
ioadvisor_set_outputload
ioadvisor_set_slew
Designer Tcl Command Reference

ioadvisor_commit (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; saves all changes in the I/O Advisor.

```
ioadvisor_commit
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following code saves all changes in the I/O Advisor:

```
ioadvisor_commit
```

See Also

ioadvisor_apply_suggestion
ioadvisor_restore
ioadvisor_restore_initial_value

ioadvisor_set_outdrive
ioadvisor_set_outputload
ioadvisor_set_slew
Designer Tcl Command Reference

ioadvisor_restore (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; restores the I/O Advisor to the initial state. All changes not committed will be lost.

```
ioadvisor_restore
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following code restores the I/O Advisor to the initial state:

```
ioadvisor_restore
```

See Also

ioadvisor_apply_suggestion
ioadvisor_commit
ioadvisor_restore_initial_value
ioadvisor_set_outdrive
ioadvisor_set_outputload
ioadvisor_set_slew
Designer Tcl Command Reference

ioadvisor_restore_initial_value (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; sets the current value for the selected attribute and I/Os to the initial value.

```
ioadvisor_restore_initial_value -attribute {value} -io {value}
```

Arguments

`-attribute{value}`

This specifies the attribute for which the values will be restored. The following table shows the acceptable values for this argument:

Value	Description
outdrive	Restores initial outdrive values
output_load	Restores initial output load values
slew	Restores initial slew values

`-io {value}`

This selects the I/Os for which the initial values will be restored. To select multiple I/Os, use `-io {value}` for each I/O.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following code restores the initial outdrive values for two I/Os.

```
ioadvisor_restore_initial_value -attribute{outdrive} -io{nPWM_out_pad} -io{PWM_out_pad}
```

See Also

ioadvisor_apply_suggestion
ioadvisor_commit
ioadvisor_restore
ioadvisor_set_outdrive
ioadvisor_set_outputload
ioadvisor_set_slew
Designer Tcl Command Reference

ioadvisor_set_outdrive (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; sets the outdrive for the selected I/Os.

```
ioadvisor_set_outdrive -io {value} -outdrive {value}
```

Arguments

`-io {value}`

This selects the I/Os for which the outdrive will be set. To select multiple I/Os, use `-io {value}` for each I/O.

`-outdrive {value}`

This specifies the outdrive for the selected I/Os.The outdrive must be a positive integer value within the list of possible outdrives of the I/Os.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following code sets the outdrive for two I/Os.

```
ioadvisor_set_outdrive -io{nPWM_out_pad} -io{PWM_out_pad} -outdrive{5}
```

See Also

ioadvisor_apply_suggestion
ioadvisor_commit
ioadvisor_restore
ioadvisor_restore_initial_value
ioadvisor_set_outputload
ioadvisor_set_slew
Designer Tcl Command Reference

ioadvisor_set_slew (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; sets the slew for the selected I/Os.

```
ioadvisor_set_slew -io {value} -slew {value}
```

Arguments

`-io {value}`

This selects the I/Os for which the slew will be set. To select multiple I/Os, use `-io {value}` for each I/O.

`-set_slew {value}`

This specifies the slew for the selected I/Os.The following table shows the acceptable values for this argument:

Value	Description
high	The slew is set to high.
low	The slew is set to low. This option is not available for all I/Os.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following code sets the slew for two I/Os.

```
ioadvisor_set_slew -io{nPWM_out_pad} -io{PWM_out_pad} -slew{high}
```

See Also

ioadvisor_apply_suggestion
ioadvisor_commit
ioadvisor_restore
ioadvisor_restore_initial_value
ioadvisor_set_outdrive
ioadvisor_set_outputload
Designer Tcl Command Reference

ioadvisor_set_outputload (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; sets the output load for the selected I/Os.

```
ioadvisor_set_outputload -io {value} -outload {value}
```

Arguments

`-io {value}`

This selects the I/Os for which the output load will be set. To select multiple I/Os, use `-io {value}` for each I/O.

`-outload {value}`

This specifies the output load for the selected I/Os.The output load must be a positive integer value.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following code sets the output load for two I/Os.

```
ioadvisor_set_outputload -io{nPWM_out_pad} -io{PWM_out_pad} -outload{5}
```

See Also

ioadvisor_apply_suggestion
ioadvisor_commit
ioadvisor_restore
ioadvisor_restore_initial_value
ioadvisor_set_outdrive
ioadvisor_set_slew
Designer Tcl Command Reference

is_source_file_current (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; audits the source file and determines whether or not the file is out of date / imported into the workspace. Returns '0' if file_name is out of date or has not been imported into the workspace, and returns '1' if file_name is current.

This command ignores the Audit settings in your ADB file.

```
is_source_file_current(filename)
```

Arguments

filename is the path to the source file

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following code determines whether or not the file has been imported into the workspace.

```
is_source_file_current (./hdl/adder.vhd)
```

See Also

are_all_source_files_curent
get_out_of_date_files
Designer Tcl Command Reference

list_clocks

Tcl command; returns details about all of the clock constraints in the current timing constraint scenario.

```
list_clocks
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_clocks]
```

See Also

create_clock
remove_clock
Tcl documentation conventions
Designer Tcl Command Reference

list_clock_latencies

Tcl command; returns details about all of the clock latencies in the current timing constraint scenario.

```
list_clock_latencies
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_clock_latencies]
```

See Also

set_clock_latency
remove_clock_latency
Tcl documentation conventions
Designer Tcl Command Reference

list_false_paths

Tcl command; returns details about all of the false paths in the current timing constraint scenario.

```
list_false_paths
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_false_paths]
```

See Also

set_false_path

`remove_false_path`

Tcl documentation conventions

Designer Tcl Command Reference

list_generated_clocks

Tcl command; returns details about all of the generated clock constraints in the current timing constraint scenario.

```
list_generated_clocks
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_generated_clocks]
```

See Also

`create_generated_clock`

`remove_generated_clock`

Tcl documentation conventions

Designer Tcl Command Reference

list_input_delays

Tcl command; returns details about all of the input delay constraints in the current timing constraint scenario.

```
list_input_delays
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_input_delays]
```

See Also

`set_input_delay`

`remove_input_delay`

Tcl documentation conventions

Designer Tcl Command Reference

list_max_delays

Tcl command; returns details about all of the maximum delay constraints in the current timing constraint scenario.

```
list_max_delays
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_max_delays]
```

See Also

set_max_delay
remove_max_delay
Tcl documentation conventions
Designer Tcl Command Reference

list_min_delays

Tcl command; returns details about all of the minimum delay constraints in the current timing constraint scenario.

```
list_min_delays
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_min_delays]
```

See Also

set_min_delay
remove_min_delay
Tcl documentation conventions
Designer Tcl Command Reference

list_multicycle_paths

Tcl command; returns details about all of the multicycle paths in the current timing constraint scenario.

```
list_multicycle_paths
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_multicycle_paths]
```

See Also

set_multicycle_path
remove_multicycle_path
Tcl documentation conventions
Designer Tcl Command Reference

list_output_delays

Tcl command; returns details about all of the output delay constraints in the current timing constraint scenario.

```
list_output_delays
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_output_delays]
```

See Also

set_output_delay
remove_output_delay
Tcl documentation conventions
Designer Tcl Command Reference

new_design (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; creates a new design. You need all three arguments for this command. This command will set up the Designer software for importing design source files

```
new_design -name design_name -family family_name –path  pathname–block value
```

Arguments

`-name design_name`

The name of the design. This is used as the base name for most of the files generated from Designer.

`-family family_name`

The Microsemi SoC device family for which the design is being targeted.

`-path path_name`

The physical path of the directory in which the design files will be created.

`block` *`value`*

Enables or disables Block mode.The following table shows the acceptable values for this option:

Value	Description
on	Enables Block mode
off	Disables Block mode

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Example 1: Creates a new ACT3 design with the name “test” in the current folder.

```
new_design -name "test" -family "ACT3" -path {.}
```

Example 2: These set of commands create a new design through variable substitution.

```
set desName “test
set famName “ACT3”
set path {d:/examples/test}
new_design -name $desName -family $famName -path $path
```

Example 3: Design creation and catch failures

```
if  { [catch { new_design -name $desName -family $famName -path $path }] {
              Puts “Failed to create a new design”
              # Handle Failure
} else {
              puts “New design creation successful”
              # Proceed to Import source files
}
```

See Also

`close_design`
`open_design`
`save_design`
`set_design`
Designer Tcl Command Reference

open_design (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; opens an existing design into the Designer software.

```
open_design file_name
```

Note: All previously open designs must be closed before opening a new design.

Arguments

`file_name`

The complete .adb file path. If the complete path is not provided, then the directory is assumed to be the current working directory.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Example 1: Opens an existing design from the file “test.adb” in the current folder.

```
open_design {test.adb}
```

Example 2: Design creation and catch failures.

```
set designFile {d:/test/my_design.adb}
if  { [catch { open_design $designFile }] {
              Puts “Failed to open design”
              # Handle Failure
} else {
              puts “Design opened successfully”
              # Proceed to further processing
}
```

See Also

close_design

new_design

save_design

Designer Tcl Command Reference

pin_assign (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; use to either assign the named pin to the specified port or assign attributes to the specified port. This command has two syntax formats. The one you use depends on what you are trying to do. The first syntax format assigns the named pin to the specified port . The second one assigns attributes to the specified port.

```
pin_assign [-nofix] -port portname -pin pin_number
pin_assign -port portname [-iostd value][-iothresh value][-outload value][-slew value][-
res_pull value]
```

Arguments

`-nofix`

Unlocks the pin assignment (by default, assignments are locked).

`-port` *`portname`*

Specifies the name of the port to which the pin is assigned.

`-pin` *`pin_number`*

Specifies the alphanumeric number of the pin to assign.

`-iostd` *`value`*

Sets the I/O standard for this pin. Choosing a standard allows the software to set other attributes such as the slew rate and output loading. If the voltage standard used with the I/O is not compatible with other I/Os in the I/O bank, then assigning an I/O standard to a port will invalidate its location and automatically unassign the I/O. The following table shows the acceptable values for the supported devices:

I/O Standards table

Use the I/O Standards table to see which I/O standards can be applied to each family:

I/O Standard	IGLOO	Fusion	ProASIC3
CMOS			

I/O Standard	IGLOO	Fusion	ProASIC3
CUSTOM			
GTLP25	IGLOOe only	X	ProASIC3E and ProASIC3L only
GTLP33	IGLOOe only	X	ProASIC3E and ProASIC3L only
GTL33	IGLOOe only	X	ProASIC3E and ProASIC3L only
GTL25	IGLOOe only	X	ProASIC3E and ProASIC3L only
HSTL1	IGLOOe only	X	ProASIC3E and ProASIC3L only
HSTLII	IGLOOe only	X	ProASIC3E and ProASIC3L only
LVCMOS33	X	X	X
LVCMOS25	IGLOOe only	X	X
LVCMOS25_50	X	X	X
LVCMOS18	X	X	X
LVCMOS15	X	X	X
LVCMOS12	X		ProASIC3L only
LVTTL	X	X	X
TTL	X	X	X
PCI	X	X	X
PCIX	X	X	X
SSTL2I	IGLOOe only	X	ProASIC3E and ProASIC3L only
SSTL2II	IGLOOe only	X	ProASIC3E and ProASIC3L only
SSTL3I	IGLOOe only	X	ProASIC3E and ProASIC3L only
SSTL3II	IGLOOe only	X	ProASIC3E and ProASIC3L only

Note: The LVDS and LVPECL I/O standards cannot be set through a script.

`-iothresh` *`value`*

Sets the compatible threshold level for inputs and outputs. The default I/O threshold is based upon the I/O standard. You can set the I/O Threshold independently of the I/O specification in the PinEditor tool by selecting **CUSTOM** in the I/O Standard cell. The following table shows the acceptable values for the supported devices:

Value	Description
CMOS	RTSX-S devices only. An advanced integrated circuit (IC) manufacturing process technology for logic and memory, characterized by high integration,

Value	Description
	low cost, low power, and high performance. CMOS logic uses a combination of p-type and n-type metal-oxide-semiconductor field effect transistors (MOSFETs) to implement logic gates and other digital circuits found in computers, telecommunications, and signal processing equipment.
LVTTL	(Low-Voltage TTL) A general purpose standard (EIA/JESDSA) for 3.3V applications. It uses an LVTTL input buffer and a push-pull output buffer.
PCI	A computer bus for attaching peripheral devices to a computer motherboard in a local bus. This standard supports both 33 MHz and 66 MHz PCI bus applications. It uses an LVTTL input buffer and a push-pull output buffer. With the aid of an external resistor, this I/O standard can be 5V-compliant for most families, excluding SmartFusion, IGLOO, ProASIC3 and Fusion families.

Note: The -iothresh attribute is also referred to as "Loading" in some families.

`-slew` *`value`*

Sets the output slew rate. Slew control affects only the falling edges. Rising edges are not affected. This attribute is only available for LVTTL, PCI, and PCI outputs. For LVTTL, it can either be high or low. For PCI and PCIX, it can only be set to high. The following table shows the acceptable values for the supported devices (SmartFusion, IGLOO, ProASIC3, Fusion):

Value	Description
high	Sets the I/O slew to high
low	Sets the I/O slew to low

`-res_pull` *`value`*

Allows you to include a weak resistor for either pull-up or pull-down of the input buffer. The following table shows the acceptable values for the supported devices (SmartFusion, IGLOO, ProASIC3, Fusion):

Value	Description
up	Includes a weak resistor for pull-up of the input buffer
down	Includes a weak resistor for pull-down of the input buffer
none	Does not include a weak resistor

`-out_load` *`value`*

Indicates the output-capacitance value based on the I/O standard selected. This option is not available in software. This attribute determines what Timer will use as the loading on the output pin and applies only to outputs. You can enter a capacitive load as an integral number of picofarads (*pF*). The default is 35*pF*. This attribute is available only for the following devices: SmartFusion, ProASIC3, Fusion.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

You must use `pin_commit` after the pin_assign command to save the changes to your design:

```
pin_assign -port usw0 -pin A2
pin_commit

pin_assign -port usw0 -iostd LVTTL -slew low -res_pull down
pin_commit
```

Note: To use a name with special characters such as square brackets [ ], you must put the entire name between curly braces { } or put a slash character \ immediately before each square bracket as shown in the following examples.

Note: The following example shows a port name enclosed with curly braces:

Note: The next example shows each square bracket preceded by a slash:

```
pin_assign -port LFSR_OUT\[15\] -iostd lvttl -slew High
```

See Also

pin_commit
pin_fix
pin_unassign
Tcl documentation conventions
Designer Tcl Command Reference

pin_commit

Tcl command; saves the pin assignments to the design (.adb) file.

```
pin_commit
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

To save pin assignments in your design, you must add the pin_commit command to the end of the script:

```
pin_commit
```

See Also

pin_fix
pin_unfix
pin_assign
pin_unassign
Tcl documentation conventions
Designer Tcl Command Reference

pin_fix (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; locks the pin assignment for the specified port, so the pins cannot be moved during place-and-route.

```
pin_fix -port portname
```

Arguments

`-port portname`

Specifies the name of the port to which the pin must be locked at its assigned location.

Note: You can assign only one pin to a port

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Fixed pins are locked pins. You cannot move locked pins during place-and-route.

Examples

You must use `pin_commit` after the pin_fix command to save the changes to your design:

```
pin_fix –port clk
pin_commit
```

See Also

`pin_commit`
`pin_unfix`
`pin_assign`
`pin_unassign`
Tcl documentation conventions
Designer Tcl Command Reference

pin_fix_all

Tcl command; locks all the assigned pins on the device so they cannot be moved during place-and-route.

```
pin_fix_all
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Fixed pins are locked pins. This command locks all the pins in your design. You cannot move locked pins during place-and-route.

Example

You must use `pin_commit` after the pin_fix_all command to save the changes to your design:

```
pin_fix_all
```

```
pin_commit
```

See Also

pin_commit
pin_fix
pin_unfix
pin_assign
pin_unassign
Tcl documentation conventions
Designer Tcl Command Reference

pin_unassign

Tcl command; unassigns the pin from the specified port. The unassigned pin location is then available for other ports. (Only one pin can be assigned to a port.)

```
pin_unassign -port portname
```

Arguments

`-port portname`

Specifies the name of the port for which the pin must be unassigned.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

You must use `pin_commit` after the pin_assign command to save the changes to your design:

```
pin_unassign –port “clk”
pin_commit
```

See Also

pin_commit
pin_fix
pin_fix_all
pin_unfix
pin_assign
pin_unassign
Tcl documentation conventions
Designer Tcl Command Reference

pin_unassign_all (SmartFusion, IGLOO, ProASIC3, and Fusion)

Tcl command; unassigns all the pins from all the ports so that all pin locations are available for assignment.

```
pin_unassign_all
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

You must use `pin_commit` after the pin_assign_all command to save the changes to your design:

```
pin_unassign_all
pin_commit
```

See Also

pin_commit
pin_fix
pin_unfix
pin_assign
pin_unassign
Tcl documentation conventions
Designer Tcl Command Reference

run_designer

Tcl command; runs Designer with compile and layout options (if selected).

```
run_designer [-logfile filename] [-script filename] [-append_commands commands] [-adb value]
[-compile value] [-layout value] [-export_ba value]
```

Arguments

`-logfile filename`

Specifies the filename of your logfile.

`-script filename`

Appends any scripts you wish to add to add to the flow, where filename is the name of the script.

`-append_commands commands`

Appends commands (if any), where commands is the list of appended commands.

`-adb value`

Creates or opens your ADB file. The following table shows the acceptable values for this argument:

Value	Description
new	Creates a new ADB file
open	Opens an existing ADB file
default	Uses the default ADB file in your Libero SoC project

`-compile value`

Compiles your design. The following table shows the acceptable values for this argument:

Value	Description
TRUE	Runs compile
FALSE	Does not run compile, proceeds to the next command

`-layout value`

Runs layout on your design. The following table shows the acceptable values for this argument:

Value	Description
TRUE	Runs layout
FALSE	Does not run layout, proceeds to the next command

`-export_ba` *`value`*

Exports back-annotate files for your design. The following table shows the acceptable values for this argument:

Value	Description
TRUE	Exports back-annotate files
FALSE	Does not export back-annotate files; proceeds to the next command

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following example creates a logfile named designerlog2 and runs compile and layout on the default ADB file created in your Libero SoC project:

```
run_designer [-logfile designerlog2] [-adb default] [-compile TRUE] [-layout TRUE]
```

See Also

Project Manager Tcl Command Reference

run_drc

Tcl command; runs the design rule check on your netlist and generates an HDL file.

```
run_drc [-netlist file] [-gen_hdl value]
```

Arguments

`-netlist` *`file`*

Name of the netlist file you want the design rule check to evaluate.

`-gen_hdl` *`value`*

Generates an HDL file (if TRUE). The following table shows the acceptable values for this argument:

Value	Description
TRUE	Generates an HDL file for your design
FALSE	Does not generate an HDL file after the design rule check

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Run the design rule check on the netlist named 'dsnr3' and generates the HDL file

```
run_drc [-netlist 'dsnr3'] [-gen_hdl TRUE]
```

See Also

Project Manager Tcl Command Reference

run_tool (SmartFusion, IGLOO, ProASIC3 and Fusion)

run_tool starts the specified tool. For tools that support command files, an optional command file can be supplied through the -script parameter.

```
run_tool
-name {tool_name}
[-script {absolute or relative path to script file}]
```

#-script is an optional parameter

```
tool_name ::= SYNTHESIZE | COMPILE | SIM_PRESYNTH | SIM_POSTSYNTH | SIM_POSTLAYOUT |
PLACEROUTE | VERIFYTIMING | VERIFYPOWER | GENERATEPROGRAMMINGFILE
|GENERATEPROGRAMMINGDATA | EXPORTPIN | EXPORTSDF | EXPORTIBIS | EXPORTPROGRAMMINGFILE |
PROGRAMDEVICE | CONFIGURE_CHAIN | PUBLISHBLOCK | SOFTWAREIDE
```

Return

run_tool returns 0 on success and 1 on failure.

Supppрted tool_names

The following table lists tool_names for run_tool –name {*tool_name*}.

tool_name	Parameter	Description
SYNTHESIZE	-script {*script_file*}	Runs synthesis on your design.
COMPILE	N/A	Runs Compile with default or configured settings.
SIM_PRESYNTH	N/A	Runs pre-synthesis simulation with your default simulation tool
SIM_POSTSYNTH	N/A	Runs post-synthesis simulation with your default simulation tool.
SIM_POSTLAYOUT	N/A	Runs post-layout simulation with your default simulation tool.
PLACEROUTE	N/A	Runs Layout with default or configured settings.
VERIFYTIMING	-script {*script_file*}	Runs timing analysis with default settings/configured settings in *script_file*.
VERIFYPOWER	-script {*script_file*}	Runs power analysis with default settings/configured settings in *script_file*.

tool_name	Parameter	Description
EXPORTPIN	N/A	Exports the pin report file. Executed after the command configure_tool –name {EXPORTPIN}..
EXPORTSDF	N/A	Exports the *_ba.sdf delay file and backannotated *_ba.v (Verilog) or *_ba.vhd (VHDL) file.
EXPORTIBIS	N/A	Exports the I/O buffer Information Specifications (IBIS) file.
GENERATEPROGRAMMINGDATA	N/A	
PROGRAMDEVICE	N/A	Programs your device with configured parameters.
EXPORTPROGRAMMINGFILE	N/A	Takes a script that contains FlashPro-specific Tcl commands and passes them to FlashPro Express for execution.
PUBLISHBLOCK	N/A	Runs SoftConsole, IAR or Keil, whichever is selected in the tool profile as the IDE tool.
SOFTWAREIDE	N/A	

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
run_tool -name {COMPILE}
run_tool \
  -name {SYNTHESIZE} -script {./control_synopsys.tcl}
  # control _synopsys.tcl contains the synthesis-specific Tcl commands
run_tool \
  -name {VERIFYTIMING} \
  -script {./SmartTime.tcl}
  # Script file containing SmartTime-specific Tcl commands
run_tool \
  -name {VERIFYPOWER} \
  -script {./SmartPower.tcl}
  # Script file containing SmartPower-specific Tcl commands
```

Note

Where possible, the value of <tool_name> corresponds to the name of the tool in Libero SoC.

Invoking some tools will cause Libero SoC to automatically run some upstream tools in the design flow. For example, if you run COMPILE, Libero runs synthesis first (as with the SYNTHESIZE command) before it runs COMPILE.

See Also

Project Manager Tcl Command Reference

save_design

Tcl command; the save_design command saves the current design in Designer to a file. If filename is not a complete path name, the ADB file is written into the current working directory.

```
save_design filename
```

Arguments

The design is written to a file denoted by the variable filename as an ADB file.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Example 1: Saves the design to a file “test.adb” in the current folder.

```
save_design {test.adb}
```

Example 2: Save design and check if it saved successfully.

```
set designFile {d:/test/my_design.adb}
if  { [catch { save_design $designFile }] {
              Puts “Failed to save design”
              # Handle Failure
} else {
              puts “Design saved successfully”
              # Proceed to make further changes
}
```

See Also

close_design
new_design
open_design
Designer Tcl Command Reference

timer_get_clock_constraints

Returns the constraints (period, frequency, and duty cycle) on the specified clock.

```
timer_get_clock_constraints -clock clock_name
```

Arguments

`-clock clock_name`

Specifies the name of the clock with the constraint to display.

Supported Families

All

Exceptions

None

Examples

The following example displays the clock constraints on the clock clk in the Log window:

```
timer_get_clock_constraints -clock clk
```

See Also

timer_get_clock_actuals

Tcl documentation conventions

timer_get_clock_actuals

Displays the actual clock frequency in the Log window, when the timing analysis tool is initiated.

```
timer_get_clock_actuals -clock clock_name
```

Arguments

`-clock clock_name`

Specifies the name of the clock with the frequency (or period) to display.

Supported Families

All

Exceptions

None

Examples

This example displays the actual clock frequency of clock clk1 in the Log window:

```
timer_get_clock_actuals -clock clk1
```

See Also

timer_get_clock_constraints

Tcl documentation conventions

timer_get_maxdelay

Displays the maximum delay constraint between two pins in a path in the Log window.

```
timer_get_maxdelay -from source_pin -to destination_pin
```

Arguments

`-from source_pin`

Specifies the name of the source pin in the path.

`-to destination_pin`

Specifies the name of the destination pin in the path.

Supported Families

All

Exceptions

None

Examples

The following example displays the maximum delay constraint from the pin clk166 to the pin reg_q_a_9_/U0:CLK in the Log window:

```
timer_get_maxdelay -from {clk166} -to {reg_q_a_9_/U0:CLK}
```

See Also

timer_set_maxdelay

Tcl documentation conventions

timer_get_path

Displays the path between the specified pins in the Log window.

```
timer_get_path -from source_pin -to destination_pin\[value ] \ [-sort value ] \ [-order value]
\ [-case value ] \ [-maxpath maximum_paths ] \ [-maxexpath maximum_paths_to_expand] \ [-
mindelay minimum_delay] \ [-maxdelay maximum_delay] \ [-breakatclk value ] \ [-breakatclr
value ]
```

Arguments

`-from source_pin`

Specifies the name of the source pin for the path.

`-to destination_pin`

Specifies the name of the destination pin for the path.

`-exp value`

Specifies whether to expand the path. The following table shows the acceptable values for this argument:

Value	Description
yes	Expands the path
no	Does not expand the path

`-sort value`

Specifies whether to sort the path by either the actual delay or slack value. The following table shows the acceptable values for this argument:

Value	Description
actual	Sorts the path by the actual delay value
slack	Sorts the path by the slack value

`-order value`

Specifies whether the maximum list size is based on the longest or shortest paths. The following table shows the acceptable values for this argument:

Value	Description
long	Base the maximum list size on the longest path in the design

Value	Description
short	Base the maximum list size on the shortest path in the design

`-case` *value*

Specifies whether the report will include timing values for the worst, typical, or best cases. The following table shows the acceptable values for this argument:

Value	Description
worst	Includes timing values for the worst cases
typ	Includes timing values for typical cases
best	Includes timing values for the best cases

`-maxpath` *maximum_paths*

Specifies the maximum number of paths to display.

`-maxexpath` *maximum_paths_to_expand*

Specifies the maximum number of paths to expand.

`-mindelay` *minimum_delay*

Specifies the path delay in the minimum delay analysis mode.

`-maxdelay` *maximum_delay*

Specifies the path delay in the maximum delay analysis mode.

`-breakatclk` *value*

Specifies whether to break the paths at the register clock pins. The following table shows the acceptable values for this argument:

Value	Description
yes	Breaks the paths at the register clock pins
no	Does not break the paths at the register clock pins

`-breakatclr` *value*

Specifies whether to break the paths at the register clear pins. The following table shows the acceptable values for this argument:

Value	Description
yes	Breaks the paths at the register clear pins
no	Does not break the paths at the register clear pins

Supported Families

All

Exceptions

None

Examples

The following example returns the paths from input port headdr_dat<31> to the input pin of register u0_headdr_data1_reg/data_out_t_31 under typical conditions.

```
timer_get_path -from "headdr_dat<31>" \
-to  "u0_headdr_data1_reg/data_out_t_31/U0:D" \
-case typ  \
-exp "yes" \
-maxpath "100" \
-maxexpapth "10"
```

The following example returns the paths from the clock pin of register gearbox_inst/bits64_out_reg<55> to the output port pma_tx_data_64bit[55]

```
timer_get_path -from "gearbox_inst/bits64_out_reg<55>/U0:CLK" \
 -to {pma_tx_data_64bit[55]} \
 -exp "yes"
```

See Also

Tcl documentation conventions

timer_get_path_constraints

Displays the path constraints that were set as the maximum delay constraint in the timing analysis tool.

```
timer_get_path_constraints
```

Arguments

None

Supported Families

All

Description

This command lists the paths constrained by maximum delay values. The information is displayed in the Log window. If no maximum delay constraints are set, this command does not report anything.

Exceptions

None

Examples

Invoking timer_get_path_constraints displays the following paths and their delay constraints in the Log window:

```
max_delay -from [all_inputs] -to [all_outputs] = 12 ns
max_delay -from p_f_testwdata0 p_f_testwdata1 -to p_f_dacuwdata0 p_f_dacuwdata1
r_f_dacuwdata0 r_f_dacuwdata1 = 8 ns
```

See Also

timer_set_maxdelay

Tcl documentation conventions

timer_remove_all_constraints

Removes all timing constraints in the current design.

```
timer_remove_all_constraints
```

Arguments

None

Supported Families

All

Exceptions

None

Examples

The following example removes all of the constraints from the design and then commits the changes:

```
timer_remove_all_constraints
timer_commit
```

See Also

timer_commit

Tcl documentation conventions

timer_remove_stop

Removes the previously entered path stop constraint on the specified pin.

```
timer_remove_stop -pin pin_name
```

Arguments

`-pin` *`pin_name`*

Specifies the name of the pin from which to remove the path stop constraint.

Supported Families

All

Description

If you remove a path stop constraint using the Timer GUI, and then export a script using **File > Export** > **Script files**, the resulting script will contain `timer_remove_pass -pin` *`pin_name`* instead of *`timer_remove_stop`* `-pin` *`pin_name`*.

Exceptions

- For the SmartFusion2, SmartFusion, IGLOO, ProASIC3, Fusion families, best practice is to use the following flow:
    1. Open **SmartTime** > **Set False Path Constraint dialog box**.
    2. Look for the pin name in the **Through:** list (Note: You must not have any entry selected in the **From** or **To** lists).
    3. Delete this pin.

Examples

The following example removes the path stop constraint on the clear pin reg_q_a_0_:CLR:

```
timer_remove_stop -pin {reg_q_a_0_:CLR}
```

See Also

timer_add_stop

Tcl documentation conventions

SmartTime User's Guide: Set False Path Constraint dialog box

timer_restore

Restores constraints previously committed in Timer.

```
timer_restore
```

Arguments

None

Supported Families

All

Exceptions

None

Examples

```
timer_restore
```

See Also

timer_commit

Tcl documentation conventions

SmartFusion, IGLOO, ProASIC3, and Fusion – Command Tools

COMPILE (SmartFusion, IGLOO, ProASIC3, and Fusion)

COMPILE is a command tool used in configure_tool and run_tool. Configure_tool allows you to configure the tool's parameters and values prior to executing the tool. Run_tool executes the tool with the configured parameters.

To compile the design in Libero SoC, you must first configure the compile tool with the configure_tool command and then execute the COMPILE command with the run_tool command.

```
configure_tool -name {COMPILE}
-params {name:value}
[-params {name:value}]
run_tool -name {COMPILE}
```

The following tables list the parameter names and values.

configure_tool –name {COMPILE} parameter:value pair

Name	Value	Description
MERGEPDC	Boolean {true | false | 1 | 0}	Set to true or 1 to merge PDC file(s) with existing physical constraints. Default is false or 0.
MERGESDC	Boolean {true | false | 1 | 0}	Set to true or 1 to merge SDC file(s) with existing timing constraints. Default is false or 0.

run_tool –name {COMPILE} Parameter:value pair

Name	Value	Description
NONE		

Supported Families

SmartFusion, IGLOO, ProASIC3, and Fusion

Example

```
configure_tool -name {COMPILE}
  -params {MERGEPDC:true}
  -params {MERGESDC:true}
run_tool –name {COMPILE} #Takes no parameters
```

Return

`configure_tool -name {COMPILE}`

Returns 0 on success and 1 on failure.

`run_tool -name {COMPILE}`

Returns 0 on success and 1 on failure.

EXPORTIBIS (SmartFusion, IGLOO, ProASIC3, Fusion)

EXPORTIBIS is a command tool used in the run_tool Tcl command to export the I/O buffer Information Specification (IBIS) file.

The IBIS (*.ibs) file is exported to <project_folder>/designer/impl1/<design_name>.ibs.

```
run_tool –name {EXPORTIBIS}
```

This command takes no parameters.

Return

Returns 0 on success and 1 on failure.

Supported Families

SmartFusion, IGLOO, ProASIC3, and Fusion

EXPORTPIN (SmartFusion, IGLOO, ProASIC3, and Fusion)

EXPORTPIN is a command tool used in the configure_tool and run_tool Tcl commands. Configure_tool allows you to configure the tool's parameters and values prior to executing the tool. The run_tool command executes the tool with the configured parameters.

To use Libero SoC's batch mode to export the pin report file of your design, you must first configure the EXPORTPIN tool with configure_tool command and then execute the EXPORTPIN command with the run_tool command.

```
configure_tool -name {EXPORTPIN}
-params {name:value}
-params {name:value}
run_tool -name {EXPORTPIN}
```

The following tables list the parameter names and values.

configure_tool –name {EXPORTPIN} parameter:value pair

Name	Value	Description
PINRPT_BY_NAME	Boolean {true | false | 1 | 0}	Set to true or 1 to have the pin report sorted by name. Default is 1.
PINRPT_BY_NUMBER	Boolean {true | false | 1 | 0}	Set to true or 1 to have the pin report sorted by package pin number. Default is false or 1.

run_tool –name {EXPORTPIN} Parameter:value pair

Name	Value	Description
NONE		

Supported Families

SmartFusion, IGLOO, ProASIC3, and Fusion

Example

```
configure_tool -name {EXPORTPIN}\
  -params {PINRPT_BY_NAME:true}\
  -params {PINRPT_BY_NUMBER:false}\
run_tool –name {EXPORTPIN} #Takes no parameters
```

Return

`configure_tool -name {EXPORTPIN}`

Returns 0 on success and 1 on failure.

`run_tool -name {EXPORTPIN}`

Returns 0 on success and 1 on failure.

EXPORTPROGRAMMINGFILE (SmartFusion, IGLOO, ProASIC3, and Fusion)

EXPORTPROGRAMMINGFILE is a command tool used in the run_tool command. The run_tool -name {EXPORTPROGRAMMINGFILE} Tcl command runs FlashPro in batch mode. It exports a STAPL file with all features available in the design. STAPL is the default file type for FlashPro. If you want to export file types other than STAPL, you must use the FlashPro GUI to specify the exported file type first before running this Tcl command in batch mode.

Refer to Export Programming File for details.

```
run_tool -name {EXPORTPROGRAMMINGFILE}
```

This command takes no parameters.

Return

Returns 0 on success and 1 on failure.

Supported Families

SmartFusion, IGLOO, ProASIC3, and Fusion

EXPORTSDF (SmartFusion, IGLOO, ProASIC3, and Fusion)

EXPORTSDF is a command tool used in the run_tool Tcl command to export the backannotated files for postlayout simulation: <design_name>_ba.sdf and <design_name>_ba.v or <design_name>_ba.vhd.

The backannotated files are exported to <project_folder>/designer/impl1/<design_name>_ba.sdf and <project_folder>/designer/impl1/<design_name>_ba.v (for Verilog project) or <project_folder>/designer/impl1/<design_name>_ba.vhd (for VHDL project)

```
run_tool –name {EXPORTSDF}
```

This command takes no parameters.

Return

Returns 0 on success and 1 on failure.

Supported Families

SmartFusion, IGLOO, ProASIC3, and Fusion

GENERATEPROGRAMMINGDATA (SmartFusion, IGLOO, ProASIC3, and Fusion)

GENERATEPROGRAMMINGDATA is the name of a command tool used in the run_tool command. The run_tool -name {GENERATEPROGRAMMINGDATA} Tcl command generates the necessary files needed for generating programming bitstream files.

```
run_tool -name {GENERATEPROGRAMMINGDATA}
```

This command takes no parameters.

Return

Returns 0 on success and 1 on failure.

Supported Families

SmartFusion, IGLOO, ProASIC3, and Fusion

PLACEROUTE (SmartFusion, IGLOO, ProASIC3, and Fusion)

PLACEROUTE is a command tool used in configure_tool and run_tool. Configure_tool allows you to configure the tool's parameters and values prior to executing the tool. Run_tool executes the tool with the configured parameters.

To compile the design in Libero SoC, you must first configure the compile tool with the configure_tool command and then execute the PLACEROUTE command with the run_tool command.

```
configure_tool -name {PLACEROUTE}
-params {name:value}
-params {name:value}
run_tool -name {PLACEROUTE}
```

The following tables list the parameter names and values.

configure_tool –name {PLACEROUTE} parameter:value pair

Name	Value	Description
TDPR	Boolean {true | false | 1 | 0}	Set to true or 1 to enable Timing-Driven Place and Route. Default is 1.
INCRPLACEANDROUTE	Boolean {true | false | 1 | 0}	Set to true or 1 to enable Incremental Place and Route. Default is false or 0.

run_tool –name {PLACEROUTE} Parameter:value pair

Name	Value	Description
NONE		

Supported Families

SmartFusion, IGLOO, ProASIC3, and Fusion

Example

```
configure_tool -name {PLACEROUTE}\
  -params {TDPR:true}\
  -params {INCRPLACEANDROUTE:true}\
run_tool –name {PLACEROUTE} #Takes no parameters
```

Return

`configure_tool -name {PLACEROUTE}`

Returns 0 on success and 1 on failure.

`run_tool -name {PLACEROUTE}`

Returns 0 on success and 1 on failure.

PUBLISHBLOCK (SmartFusion, IGLOO, ProASIC3, and Fusion)

PUBLISHBLOCK is a command tool used in configure_tool and run_tool. Configure_tool allows you to configure the tool's parameters and values prior to executing the tool. Run_tool executes the PUBLISHBLOCK command tool with the configured parameters.

From a project with Block Design Flow enabled, you can publish block information which can then be imported into the top level project as part of the bottom-up design methodology. To publish block information, you must first configure the PUBLISHBLOCK tool with the configure_tool command and then execute the PUBLISHBLOCK command with the run_tool command.

```
configure_tool -name {PUBLISHBLOCK}
-params {name:value}
-params {name:value}
-params {name:value}
-params {name:value}
run_tool -name {PUBLISHBLOCK}
```

configure_tool –name {PUBLISHBLOCK} parameter:value pair

Params_name	<Params_value>	Description
LANGUAGE	String {verilog | vhdl}	Set to Verilog or VHDL as the Block Hardware Description Language. The default is the preferred HDL type set in your project setting.
PLACEMENT	Boolean {true | false | 1 | 0}	Set to true or 1 to publish placement information for the block. Default is false or 0.
ROUTING	Boolean {true | false | 1 | 0}	Set to true or 1 to retain the region constraint information with the block. Default is false or 0.
REGION	Boolean {true | false | 1 | 0}	Set to true or 1 to retain the region constraint information with the block. Default is false or 0.

Run_tool –name {PUBLISHBLOCK} Parameter:value pair

Params_name	<Params_value>	Description

Params_name	<Params_value>	Description
NONE		

Supported Families

SmartFusion, IGLOO, ProASIC3, and Fusion

Example

```
configure_tool -name {PUBLISHBLOCK}\
  -params {LANGUAGE_VERILOG_2001:true}\
  -params {PLACEMENT:true}\
  -params {ROUTING:true}\
  -params {REGION:true}
run_tool –name {PUBLISHBLOCK} #Takes no parameters
```

Return

`configure_tool -name {PUBLISHBLOCK}`

Returns 0 on success and 1 on failure.

`run_tool -name {PUBLISHBLOCK}`

Returns 0 on success and 1 on failure.

SYNTHESIZE

SYNTHESIZE is a command tool used in configure_tool and run_tool. Configure_tool is a general-purpose Tcl command that allows you to configure a tool's parameters and values prior to executing the tool. The run_tool Tcl command then executes the specified tool with the configured parameters.

To synthesize your design in Libero SoC, you first configure the synthesize tool with the configure_tool command and then execute the command with the run_tool command.

```
configure_tool -name {SYNTHESIZE}
-params {name:value}
run_tool -name {SYNTHESIZE}
```

The following tables list the parameter names and values.

configure_tool –name {SYNTHESIZE} parameter:value pair

Name	Value	Description
LANGUAGE_VERILOG_2001	Boolean {true | false | 1 | 0}	Set to true or 1 when the input HDL language is Verilog 2001.
LANGUAGE_SYSTEM_VLOG	Boolean {true | false | 1 | 0}	Set to true or 1 when the input HDL language is System Verilog.
LANGUAGE_VHDL_2008	Boolean {true | false | 1 | 0}	Set to true or 1 when the input HDL language is VHDL 2008.

Name	Value	Description
CLOCK_GLOBAL	Integer	Specifies the threshold value for Clock pin promotion. The default is 2.
CLOCK_DATA	Integer value between 1000 and 200,000.	Specifies the threshold value for data pin promotion. The default is 5000.
RAM_OPTIMIZED_FOR_POWER	Boolean {true | false | 1 | 0}	Set to true or 1 to optimize RAM for Low Power; RAMS are inferred and configured to ensure the lowest power consumption. Set to false or 0 to optimize RAM for High Speed at the expense of more FPGA resources.
RETIMING	Boolean {true | false | 1 | 0}	Set to true or 1 to enable Retiming during synthesis. Set to false or 0 to disable Retiming during synthesis.

Run_tool –name {SYNTHESIZE} Parameter:value pair

Name	Value	Description
NONE		

Supported Families

SmartFusion2, IGLOO2

Example

```
configure_tool -name {SYNTHESIZE}\
  -params {LANGUAGE_VERILOG_2001:true}
run_tool –name {SYNTHESIZE} #Takes no parameters
```

Return

`configure_tool -name {SYNTHESIZE}`

Returns 0 on success and 1 on failure.

`run_tool -name {SYNTHESIZE}`

Returns 0 on success and 1 on failure.

VERIFYTIMING (SmartFusion, IGLOO, ProASIC3, and Fusion)

VERIFYTIMING is a command tool used in run_tool. Run_tool executes the VERIFYTIMING command. By default, the max_delay, min_delay and timing_violation reports are generated when the command is run. For more advanced features such as setting timing constraints, exporting timing constraints and

setting user paths, you must invoke Designer and execute the SmartTime-specific Tcl commands from Designer.

```
run_tool -name {VERIFYTIMING}
```

Supported Families

SmartFusion, IGLOO, ProASIC3, and Fusion

Example

```
run_tool –name {VERIFYTIMING}
```

Return

Returns 0 on success and 1 on failure.

SmartFusion2, IGLOO2, and RTG4 - SmartTime

all_outputs

Tcl command; returns an object representing all output and inout pins in the current design.

```
all_outputs
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Exceptions

You can only use this command as part of a –from, -to, or –through argument in the following Tcl commands: set_min_delay, set_max_delay, set_multicycle_path, and set_false_path.

Examples

```
set_max_delay -from [all_inputs] -to [all_outputs]
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

all_registers

Tcl command; returns an object representing register pins or cells in the current scenario based on the given parameters.

```
all_registers [-clock clock_name]
[-async_pins][-output_pins][-data_pins][-clock_pins]
```

Arguments

-clock *clock_name*

Specifies the name of the clock domain to which the registers belong. If no clock is specified, all registers in the design will be targeted.

-async_pins

Lists all register pins that are async pins for the specified clock (or all registers asynchronous pins in the design).

-output_pins

Lists all register pins that are output pins for the specified clock (or all registers output pins in the design).

-data_pins

Lists all register pins that are data pins for the specified clock (or all registers data pins in the design).

-clock_pins

Lists all register pins that are data pins for the specified clock (or all registers clock pins in the design).

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Exceptions

You can only use this command as part of a –from, -to, or –through argument in the following Tcl commands: set_min_delay, set_max_delay, set_multicycle_path, and set_false_path.

Examples

```
set_max_delay 2.000 -from { ff_m:CLK ff_s2:CLK } -to [all_registers -clock_pins -clock { ff_m:Q }]
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

check_constraints

Tcl command; checks all timing constraints in the current scenario for validity. This command performs the same checks as when the constraint is entered through SDC or Tcl.

```
check_constraints
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
check_constraints
```

clone_scenario

Tcl command; creates a new timing scenario by duplicating an existing one. You must provide a unique name (that is, it cannot already be used by another timing scenario).

```
clone_scenario name -source origin
```

Arguments

name

Specifies the name of the new timing scenario to create.

-source *origin*

Specifies the source of the timing scenario to clone (copy). The source must be a valid, existing timing scenario.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

This command creates a timing scenario with the specified name, which includes a copy of all constraints in the original scenario (specified with the -source parameter). The new scenario is then added to the list of scenarios.

Example

```
clone_scenario scenario_A –source {Primary}
```

See Also

create_scenario
delete_scenario
Tcl documentation conventions
Designer Tcl Command Reference

create_clock

Tcl command; creates a clock constraint on the specified ports/pins, or a virtual clock if no source other than a name is specified.

```
create_clock -period period_value [-name clock_name]
[-waveform> edge_list][source_objects]
```

Arguments

`-period period_value`

Specifies the clock period in nanoseconds. The value you specify is the minimum time over which the clock waveform repeats. The period_value must be greater than zero.

`-name clock_name`

Specifies the name of the clock constraint. You must specify either a clock name or a source.

`-waveform edge_list`

Specifies the rise and fall times of the clock waveform in ns over a complete clock period. There must be exactly two transitions in the list, a rising transition followed by a falling transition. You can define a clock starting with a falling edge by providing an edge list where fall time is less than rise time. If you do not specify -waveform option, the tool creates a default waveform, with a rising edge at instant 0.0 ns and a falling edge at instant (period_value/2)ns.

source_objects

Specifies the source of the clock constraint. The source can be ports, pins, or nets in the design. If you specify a clock constraint on a pin that already has a clock, the new clock replaces the existing one. You must specify either a source or a clock name.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Creates a clock in the current design at the declared source and defines its period and waveform. The static timing analysis tool uses this information to propagate the waveform across the clock network to the clock pins of all sequential elements driven by this clock source.

The clock information is also used to compute the slacks in the specified clock domain that drive optimization tools such as place-and-route.

Examples

The following example creates two clocks on ports CK1 and CK2 with a period of 6, a rising edge at 0, and a falling edge at 3:

```
create_clock -name {my_user_clock} -period 6 CK1
create_clock -name {my_other_user_clock} –period 6 –waveform {0 3} {CK2}
```

The following example creates a clock on port CK3 with a period of 7, a rising edge at 2, and a falling edge at 4:

```
create_clock –period 7 –waveform {2 4} [get_ports {CK3}]
```

See Also

create_generated_clock

Tcl Command Documentation Conventions

Designer Tcl Command Reference

create_generated_clock

Tcl command; creates an internally generated clock constraint on the ports/pins and defines its characteristics.

```
create_generated_clock [-name name] -source reference_pin [-divide_by divide_factor] [-
multiply_by multiply_factor] [-invert] source
```

Arguments

`-name` *name*

Specifies the name of the clock constraint.

`-source` *reference_pin*

Specifies the reference pin in the design from which the clock waveform is to be derived.

`-divide_by` *divide_factor*

Specifies the frequency division factor. For instance if the *divide_factor* is equal to 2, the generated clock period is twice the reference clock period.

`-multiply_by` *multiply_factor*

Specifies the frequency multiplication factor. For instance if the *multiply_factor* is equal to 2, the generated clock period is half the reference clock period.

`-invert`

Specifies that the generated clock waveform is inverted with respect to the reference clock.

`source`

Specifies the source of the clock constraint on internal pins of the design. If you specify a clock constraint on a pin that already has a clock, the new clock replaces the existing clock. Only one source is accepted. Wildcards are accepted as long as the resolution shows one pin.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Creates a generated clock in the current design at a declared source by defining its frequency with respect to the frequency at the reference pin. The static timing analysis tool uses this information to compute and propagate its waveform across the clock network to the clock pins of all sequential elements driven by this source.

The generated clock information is also used to compute the slacks in the specified clock domain that drive optimization tools such as place-and-route.

Examples

The following example creates a generated clock on pin U1/reg1:Q with a period twice as long as the period at the reference port CLK.

```
create_generated_clock -name {my_user_clock} –divide_by 2 –source [get_ports {CLK}]  U1/reg1:Q
```

The following example creates a generated clock at the primary output of myPLL with a period ¾ of the period at the reference pin clk.

```
create_generated_clock –divide_by 3 –multiply_by 4  -source clk [get_pins {myPLL:CLK1}]
```

See Also

create_clock

Tcl Command Documentation Conventions

Designer Tcl Command Reference

create_scenario

Tcl command; creates a new timing scenario with the specified name. You must provide a unique name (that is, it cannot already be used by another timing scenario).

```
create_scenario name
```

Arguments

name

Specifies the name of the new timing scenario.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

A timing scenario is a set of timing constraints used with a design. Scenarios enable you to easily refine the set of timing constraints used for Timing-Driven Place-and-Route, so as to achieve timing closure more rapidly.

This command creates an empty timing scenario with the specified name and adds it to the list of scenarios.

Example

```
create_scenario scenario_A
```

See Also

clone_scenario

Tcl Command Documentation Conventions

Designer Tcl Command Reference

get_cells

Tcl command; returns an object representing the cells (instances) that match those specified in the pattern argument.

```
get_cells pattern
```

Arguments

`pattern`

Specifies the pattern to match the instances to return. For example, "get_cells U18*" returns all instances starting with the characters "U18", where “*” is a wildcard that represents any character string.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

This command returns a collection of instances matching the pattern you specify. You can only use this command as part of a –from, -to, or –through argument in the following Tcl commands: set_max_delay, set_multicycle_path, and set_false_path.

Examples

```
set_max_delay 2 -from [get_cells {reg*}] -to [get_ports {out}]
set_false_path –through [get_cells {Rblock/muxA}]
```

See Also

get_clocks
get_nets
get_pins
get_ports
Tcl Command Documentation Conventions
Designer Tcl Command Reference

get_clocks

Tcl command; returns an object representing the clock(s) that match those specified in the pattern argument in the current timing scenario.

```
get_clocks pattern
```

Arguments

pattern

Specifies the pattern to use to match the clocks set in SmartTime or Timer.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

- If this command is used as a –from argument in either the set maximum (set_max_delay), or set minimum delay (set_min_delay), false path (set_false_path), and multicycle constraints (set_multicycle_path), the clock pins of all the registers related to this clock are used as path start points.
- If this command is used as a –to argument in either the set maximum (set_max_delay), or set minimum delay (set_min_delay), false path (set_false_path), and multicycle constraints (set_multicycle_path), the synchronous pins of all the registers related to this clock are used as path endpoints.

Example

```
set_max_delay -from [get_ports datal] -to \
[get_clocks ck1]
```

See Also

create_clock
create_generated_clock
Tcl Command Documentation Conventions
Designer Tcl Command Reference

get_current_scenario

Tcl command; returns the name of the current timing scenario.

```
get_current_scenario
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
get_current_scenario
```

See Also

set_current_scenario
Tcl documentation conventions
Designer Tcl Command Reference

get_nets

Tcl command; returns an object representing the nets that match those specified in the pattern argument.

```
get_nets pattern
```

Arguments

pattern

Specifies the pattern to match the names of the nets to return. For example, "get_nets N_255*" returns all nets starting with the characters "N_255", where “*” is a wildcard that represents any character string.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

This command returns a collection of nets matching the pattern you specify. You can only use this command as source objects in create clock (create_clock) or create generated clock (create_generated_clock) constraints and as `-through` arguments in the set false path, set minimum delay, set maximum delay, and set multicycle path constraints.

Examples

```
set_max_delay 2 -from [get_ports RDATA1] -through [get_nets {net_chkp1 net_chkqi}]
set_false_path –through [get_nets {Tblk/rm/n*}]
create_clock -name mainCLK -period 2.5 [get_nets {cknet}]
```

See Also

create_clock
create_generated_clock
set_false_path
set_min_delay
set_max_delay
set_multicycle_path
Tcl documentation conventions
Designer Tcl Command Reference

get_pins

Tcl command; returns an object representing the pin(s) that match those specified in the pattern argument.

```
get_pins pattern
```

Arguments

pattern

Specifies the pattern to match the pins to return. For example, "get_pins clock_gen*" returns all pins starting with the characters "clock_gen", where “*” is a wildcard that represents any character string.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
create_clock -period 10 [get_pins clock_gen/reg2:Q]
```

See Also

create_clock
create_generated_clock
set_clock_latency
set_false_path
set_min_delay
set_max_delay
set_multicycle_path
Tcl documentation conventions
Designer Tcl Command Reference

get_ports

Tcl command; returns an object representing the port(s) that match those specified in the pattern argument.

```
get_portspattern
```

Argument

pattern

Specifies the pattern to match the ports.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
create_clock -period 10 [get_ports CK1]
```

See Also

create_clock
set_clock_latency
set_input_delay
set_output_delay
set_min_delay
set_max_delay
set_false_path
set_multicycle_path
Tcl documentation conventions
Designer Tcl Command Reference

list_clock_latencies

Tcl command; returns details about all of the clock latencies in the current timing constraint scenario.

```
list_clock_latencies
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_clock_latencies]
```

See Also

set_clock_latency
remove_clock_latency
Tcl documentation conventions
Designer Tcl Command Reference

list_clock_uncertainties

Tcl command; returns details about all of the clock uncertainties in the current timing constraint scenario.

```
list_clock_uncertainties
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
list_clock_uncertainties
```

See Also

set_clock_uncertainty
remove_clock_uncertainty
Designer Tcl Command Reference

list_clocks

Tcl command; returns details about all of the clock constraints in the current timing constraint scenario.

```
list_clocks
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_clocks]
```

See Also

create_clock
remove_clock
Tcl documentation conventions
Designer Tcl Command Reference

list_disable_timings

Tcl command; returns the list of disable timing constraints for the current scenario.

```
list_disable_timings
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
list_disable_timings
```

See Also

Designer Tcl Command Reference

list_false_paths

Tcl command; returns details about all of the false paths in the current timing constraint scenario.

```
list_false_paths
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_false_paths]
```

See Also

`set_false_path`

`remove_false_path`

Tcl documentation conventions

Designer Tcl Command Reference

list_generated_clocks

Tcl command; returns details about all of the generated clock constraints in the current timing constraint scenario.

```
list_generated_clocks
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_generated_clocks]
```

See Also

`create_generated_clock`

`remove_generated_clock`

Tcl documentation conventions

Designer Tcl Command Reference

list_input_delays

Tcl command; returns details about all of the input delay constraints in the current timing constraint scenario.

```
list_input_delays
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_input_delays]
```

See Also

set_input_delay
remove_input_delay
Tcl documentation conventions
Designer Tcl Command Reference

list_max_delays

Tcl command; returns details about all of the maximum delay constraints in the current timing constraint scenario.

```
list_max_delays
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_max_delays]
```

See Also

set_max_delay
remove_max_delay
Tcl documentation conventions
Designer Tcl Command Reference

list_min_delays

Tcl command; returns details about all of the minimum delay constraints in the current timing constraint scenario.

```
list_min_delays
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_min_delays]
```

See Also

set_min_delay
remove_min_delay
Tcl documentation conventions
Designer Tcl Command Reference

list_multicycle_paths

Tcl command; returns details about all of the multicycle paths in the current timing constraint scenario.

```
list_multicycle_paths
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_multicycle_paths]
```

See Also

set_multicycle_path
remove_multicycle_path
Tcl documentation conventions
Designer Tcl Command Reference

list_objects

Tcl command; returns a list of object matching the parameter. Objects can be nets, pins, ports, clocks or instances.

```
list_objects <object>
```

Arguments

Any timing constraint parameter.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following example lists all the inputs in your design:

```
list_objects [all_inputs]
```

You can also use wildcards to filter your list, as in the following command:

```
list_objects [get_ports a*]
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

list_output_delays

Tcl command; returns details about all of the output delay constraints in the current timing constraint scenario.

```
list_output_delays
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
puts [list_output_delays]
```

See Also

set_output_delay

remove_output_delay

Tcl documentation conventions

Designer Tcl Command Reference

list_paths

Tcl command; returns a list of the *n* worst paths matching the arguments. The number of paths returned can be changed using the set_options -limit_max_paths <value> command.

```
list_paths
-analysis <max | min>
-format <csv | text>
-set <name>
-clock <clock name>
-type <set_type>
-from_clock <clock name>
-to_clock <clock name>
-in_to_out
-from <port/pin pattern>
-to <port/pin pattern>
```

Arguments

`-analysis <max | min>`

Specifies whether the timing analysis is done for max-delay (setup check) or min-delay (hold check). Valid values are: max or min.

`-format < text | csv >`

Specifies the list format. It can be either text (default) or csv (comma separated values). Text format is better for display and csv format is better for parsing.

`-set <name>`

Returns a list of paths from the named set. You can either use the -set option to specify a user set by its name or use both -clock and -type to specify a set.

`-clock <clock name>`

Returns a list of paths from the specified clock domain. This option requires the -type option.

`-type <set_type>`

Specifies the type of paths to be included. It can only be used along with -clock. Valid values are:

reg_to_reg -- Paths between registers

external_setup -- Path from input ports to data pins of registers

external_hold -- Path from input ports to data pins of registers

clock_to_out -- Path from registers to output ports

reg_to_async -- Path from registers to asynchronous pins of registers

external_recovery -- Path from input ports to asynchronous pins of registers

external_removal -- Path from input ports to asynchronous pins of registers

async_to_reg -- Path from asynchronous pins to registers

`-from_clock <clock name>`

Used along with -to_clock to get the list of paths of the inter-clock domain between the two clocks.

`-to_clock <clock name>`

Used along with -from_clock to get the list of paths of the inter-clock domain between the two clocks.

`-in_to_out`

Used to get the list of path between input and output ports.

`-from <port/pin pattern>`

Filter the list of paths to those starting from ports or pins matching the pattern.

`-to <port/pin pattern>`

Filter the list of paths to those ending at ports or pins matching the pattern.

Example

The following command displays the list of register to register paths of clock domain clk1:

```
puts [ list_paths -clock clk1 -type reg_to_reg ]
```

See Also

create_set

expand_path

set_options

list_scenarios

Tcl command; returns a list of names of all of the available timing scenarios.

```
list_scenarios
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

```
list_scenarios
```

See Also

get_current_scenario

Tcl documentation conventions

Designer Tcl Command Reference

read_sdc

The read_sdc Tcl command evaluate an SDC file, adding all constraints to the specified scenario (or the current/default one if none is specified). Existing constraints are removed if -add is not specified.

```
read_sdc
-add
-scenario scenario_name
-netlist (user | optimized)
-pin_separator (: | /)
file name
```

Arguments

`-add`

Specifies that the constraints from the SDC file will be added on top of the existing ones, overriding them in case of a conflict. If not used, the existing constraints are removed before the SDC file is read.

`-scenario` *`scenario_name`*

Specifies the scenario to add the constraints to. The scenario is created if none exists with this name.

`-netlist (`*`user`* | *`optimized`*`)`

Specifies whether the SDC file contains object defined at the post-synthesis netlist (user) level or physical (optimized) netlist (used for timing analysis).

`-pin_separator` *`sep`*

Specify the pin separator used in the SDC file. It can be either ':' or '/'.

`file name`

Specify the SDC file name.

Example

The following command removes all constraints from the current/default scenario and adds all constraints from design.sdc file to it:

```
read_sdc design.sdc
```

See Also

write_sdc

remove_all_constraints

Tcl command; removes all timing constraints from analysis.

```
remove_all_constraints
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
remove_all_constraints
```

See Also

remove_scenario

remove_clock

Tcl command; removes the specified clock constraint from the current timing scenario.

```
remove_clock -name clock_name| -id constraint_ID
```

Arguments

`-name clock_name`

Specifies the name of the clock constraint to remove from the current scenario. You must specify either a clock name or an ID.

`-id constraint_ID`

Specifies the ID of the clock constraint to remove from the current scenario. You must specify either an ID or a clock name that exists in the current scenario.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes the specified clock constraint from the current scenario. If the specified name does not match a clock constraint in the current scenario, or if the specified ID does not refer to a clock constraint, this command fails.

Do not specify both the name and the ID.

Exceptions

You cannot use wildcards when specifying a clock name.

Examples

The following example removes the clock constraint named "my_user_clock":

```
remove_clock -name my_user_clock
```

The following example removes the clock constraint using its ID:

```
set clockId [create_clock –name my_user_clock –period 2]
remove_clock –id $clockId
```

See Also

create_clock

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_clock_latency

Tcl command; removes a clock source latency from the specified clock and from all edges of the clock.

```
remove_clock_latency {-source clock_name_or_source |-id constraint_ID}
```

Arguments

`-source clock_name_or_source`

Specifies either the clock name or source name of the clock constraint from which to remove the clock source latency. You must specify either a clock or source name or its constraint ID.

`-id constraint_ID`

Specifies the ID of the clock constraint to remove from the current scenario. You must specify either a clock or source name or its constraint ID.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes a clock source latency from the specified clock in the current scenario. If the specified source does not match a clock with a latency constraint in the current scenario, or if the specified ID does not refer to a clock with a latency constraint, this command fails.

Do not specify both the source and the ID.

Exceptions

You cannot use wildcards when specifying a clock name.

Examples

The following example removes the clock source latency from the specified clock.

```
remove_clock_latency -source my_clock
```

See Also

`set_clock_latency`

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_clock_uncertainty

Tcl command; removes a clock-to-clock uncertainty from the current timing scenario by specifying either its exact arguments or its ID.

```
remove_clock_uncertainty -from | -rise_from | -fall_from from_clock_list -to | -rise_to| -
fall_to to_clock_list -setup {value} -hold {value}
remove_clock_uncertainty -id constraint_ID
```

Arguments

`-from`

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the source clock list. Only one of the `-from`, `-rise_from`, or `-fall_from` arguments can be specified for the constraint to be valid.

`-rise_from`

Specifies that the clock-to-clock uncertainty applies only to rising edges of the source clock list. Only one of the `-from`, `-rise_from`, or `-fall_from` arguments can be specified for the constraint to be valid.

`-fall_from`

Specifies that the clock-to-clock uncertainty applies only to falling edges of the source clock list. Only one of the `-from`, `-rise_from`, or `-fall_from` arguments can be specified for the constraint to be valid.

`from_clock_list`

Specifies the list of clock names as the uncertainty source.

`-to`

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the destination clock list. Only one of the `-to`, `-rise_to` , or `-fall_to` arguments can be specified for the constraint to be valid.

`-rise_to`

Specifies that the clock-to-clock uncertainty applies only to rising edges of the destination clock list. Only one of the `-to`, `-rise_to` , or `-fall_to` arguments can be specified for the constraint to be valid.

`-fall_to`

Specifies that the clock-to-clock uncertainty applies only to falling edges of the destination clock list. Only one of the `-to`, `-rise_to` , or `-fall_to` arguments can be specified for the constraint to be valid.

`to_clock_list`

Specifies the list of clock names as the uncertainty destination.

`-setup`

Specifies that the uncertainty applies only to setup checks. If none or both `-setup` and `-hold` are present, the uncertainty applies to both setup and hold checks.

`-hold`

Specifies that the uncertainty applies only to hold checks. If none or both `-setup` and `-hold` are present, the uncertainty applies to both setup and hold checks.

`-id` *`constraint_ID`*

Specifies the ID of the clock constraint to remove from the current scenario. You must specify either the exact parameters to set the constraint or its constraint ID.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes a clock-to-clock uncertainty from the specified clock in the current scenario. If the specified arguments do not match clocks with an uncertainty constraint in the current scenario, or if the specified ID does not refer to a clock-to-clock uncertainty constraint, this command fails.

Do not specify both the exact arguments and the ID.

Examples

```
remove_clock_uncertainty -from Clk1 -to Clk2
remove_clock_uncertainty -from Clk1 -fall_to { Clk2 Clk3 } -setup
remove_clock_uncertainty 4.3 -fall_from { Clk1 Clk2 } -rise_to *
remove_clock_uncertainty 0.1 -rise_from [ get_clocks { Clk1 Clk2 } ] -fall_to { Clk3 Clk4 } -setup
remove_clock_uncertainty 5 -rise_from Clk1 -to [ get_clocks {*} ]
remove_clock_uncertainty -id $clockId
```

See Also

`remove_clock`

`remove_generated_clock`

`set_clock_uncertainty`

Designer Tcl Command Reference

remove_disable_timing

Tcl command; removes a disable timing constraint by specifying its arguments, or its ID. If the arguments do not match a disable timing constraint, or if the ID does not refer to a disable timing constraint, the command fails.

```
remove_disable_timing -from value -to value name -id name
```

Arguments

`-from from_port`

Specifies the starting port. The –from and –to arguments must either both be present or both omitted for the constraint to be valid.

`-to to_port`

Specifies the ending port. The –from and –to arguments must either both be present or both omitted for the constraint to be valid.

name

Specifies the cell name where the disable timing constraint will be removed. It is an error to supply both a cell name and a constraint ID, as they are mutually exclusive. No wildcards are allowed when specifying a clock name, either alone or in an accessor command1.

`-id name`

Specifies the constraint name where the disable timing constraint will be removed. It is an error to supply both a cell name and a constraint ID, as they are mutually exclusive. No wildcards are allowed when specifying a clock name, either alone or in an accessor command1.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
remove_disable_timing -from port1 -to port2 -id new_constraint
```

Designer Tcl Command Reference

remove_false_path

Tcl command; removes a false path from the current timing scenario by specifying either its exact arguments or its ID.

```
remove_false_path [-from from_list] [-to to_list] [-through through_list] [-id constraint_ID]
remove_false_path -id constraint_ID
```

Arguments

`-from from_list`

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through through_list`

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

`-to to_list`

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-id constraint_ID`

Specifies the ID of the false path constraint to remove from the current scenario. You must specify either the exact false path to remove or the constraint ID that refers to the false path constraint to remove.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes a false path from the specified clock in the current scenario. If the arguments do not match a false path constraint in the current scenario, or if the specified ID does not refer to a false path constraint, this command fails.

Do not specify both the false path arguments and the constraint ID.

Exceptions

You cannot use wildcards when specifying a clock name, either alone or in an Accessor command such as get_pins or get_ports.

Examples

The following example specifies all false paths to remove:

```
remove_false_path -through U0/U1:Y
```

The following example removes the false path constraint using its id:

```
set fpId [set_false_path –from [get_clocks c*] –through {topx/reg/*} –to [get_ports out15] ]
remove_false_path –id $fpId
```

See Also

set_false_path

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_generated_clock

Tcl command; removes the specified generated clock constraint from the current scenario.

```
remove_generated_clock {-name clock_name | -id constraint_ID }
```

Arguments

`-name` *`clock_name`*

Specifies the name of the generated clock constraint to remove from the current scenario. You must specify either a clock name or an ID.

`-id` *`constraint_ID`*

Specifies the ID of the generated clock constraint to remove from the current scenario. You must specify either an ID or a clock name that exists in the current scenario.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes the specified generated clock constraint from the current scenario. If the specified name does not match a generated clock constraint in the current scenario, or if the specified ID does not refer to a generated clock constraint, this command fails.

Do not specify both the name and the ID.

Exceptions

You cannot use wildcards when specifying a generated clock name.

Examples

The following example removes the generated clock constraint named "my_user_clock":

```
remove_generated_clock -name my_user_clock
```

See Also

create_generated_clock

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_input_delay

Tcl command; removes an input delay a clock on a port by specifying both the clocks and port names or the ID of the input_delay constraint to remove.

```
remove_input_delay -clock clock_name port_pin_list
remove_input_delay -id constraint_ID
```

Arguments

`-clock` *`clock_name`*

Specifies the clock name to which the specified input delay value is assigned.

`port_pin_list`

Specifies the port names to which the specified input delay value is assigned.

`-id` *`constraint_ID`*

Specifies the ID of the clock with the input_delay value to remove from the current scenario. You must specify either both a clock name and list of port names or the input_delay constraint ID .

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes an input delay from the specified clocks and port in the current scenario. If the clocks and port names do not match an input delay constraint in the current scenario, or if the specified ID does not refer to an input delay constraint, this command fails.

Do not specify both the clock and port names and the constraint ID.

Exceptions

You cannot use wildcards when specifying a clock name, either alone or in an accessor command.

Examples

The following example removes the input delay from CLK1 on port data1:

```
remove_input_delay -clock [get_clocks CLK1] [get_ports data1]
```

See Also

set_input_delay

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_max_delay

Tcl command; removes a maximum delay constraint from the current timing scenario by specifying either its exact arguments or its ID.

```
remove_max_delay [-from from_list] [-to to_list] [-through through_list]
remove_max_delay -id constraint_ID
```

Arguments

`-from from_list`

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through through_list`

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

`-to to_list`

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-id constraint_ID`

Specifies the ID of the maximum delay constraint to remove from the current scenario. You must specify either the exact maximum delay arguments to remove or the constraint ID that refers to the maximum delay constraint to remove.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes a maximum delay value from the specified clock in the current scenario. If the arguments do not match a maximum delay constraint in the current scenario, or if the specified ID does not refer to a maximum delay constraint, this command fails.

Do not specify both the maximum delay arguments and the constraint ID.

Exceptions

You cannot use wildcards when specifying a clock name, either alone or in an Accessor command.

Examples

The following example specifies a range of maximum delay constraints to remove:

```
remove_max_delay -through U0/U1:Y
```

See Also

set_max_delay

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_min_delay

Tcl command; removes a minimum delay constraint in the current timing scenario by specifying either its exact arguments or its ID.

```
remove_min_delay [-from from_list] [-to to_list] [-through through_list]
remove_min_delay -id constraint_ID
```

Arguments

`-from` *`from_list`*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through` *`through_list`*

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

`-to`*`to_list`*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-id` *`constraint_ID`*

Specifies the ID of the minimum delay constraint to remove from the current scenario. You must specify either the exact minimum delay arguments to remove or the constraint ID that refers to the minimum delay constraint to remove.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes a minimum delay value from the specified clock in the current scenario. If the arguments do not match a minimum delay constraint in the current scenario, or if the specified ID does not refer to a minimum delay constraint, this command fails.

Do not specify both the minimum delay arguments and the constraint ID.

Exceptions

You cannot use wildcards when specifying a clock name, either alone or in an accessor command.

Examples

The following example specifies a range of minimum delay constraints to remove:

```
remove_min_delay -through U0/U1:Y
```

See Also

`set_min_delay`

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_multicycle_path

Tcl command; removes a multicycle path constraint in the current timing scenario by specifying either its exact arguments or its ID.

```
remove_multicycle_path [-from from_list] [-to to_list] [-through through_list]
remove multicycle_path -id constraint_ID
```

Arguments

`-from` *`from_list`*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through` *`through_list`*

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

`-to`*`to_list`*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-id` *`constraint_ID`*

Specifies the ID of the multicycle path constraint to remove from the current scenario. You must specify either the exact multicycle path arguments to remove or the constraint ID that refers to the multicycle path constraint to remove.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes a multicycle path from the specified clock in the current scenario. If the arguments do not match a multicycle path constraint in the current scenario, or if the specified ID does not refer to a multicycle path constraint, this command fails.

Do not specify both the multicycle path arguments and the constraint ID.

Exceptions

You cannot use wildcards when specifying a clock name, either alone or in an accessor command.

Examples

The following example removes all paths between reg1 and reg2 to 3 cycles for setup check.

```
remove_multicycle_path -from [get_pins {reg1}] –to [get_pins {reg2}]
```

See Also

`set_multicycle_path`

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_output_delay

Tcl command; removes an ouput delay by specifying both the clocks and port names or the ID of the output_delay constraint to remove.

```
remove_output_delay -clock clock_name port_pin_list
remove_output_delay -id constraint_ID
```

Arguments

`-clock` *`clock_name`*

Specifies the clock name to which the specified output delay value is assigned.

`port_pin_list`

Specifies the port names to which the specified output delay value is assigned.

`-id` *`constraint_ID`*

Specifies the ID of the clock with the output_delay value to remove from the current scenario. You must specify either both a clock name and list of port names or the output_delay constraint ID .

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Removes an output delay from the specified clocks and port in the current scenario. If the clocks and port names do not match an output delay constraint in the current scenario, or if the specified ID does not refer to an output delay constraint, this command fails.

Do not specify both the clock and port names and the constraint ID.

Exceptions

You cannot use wildcards when specifying a clock name, either alone or in an accessor command.

Examples

The following example removes the output delay from CLK1 on port out1:

```
remove_output_delay -clock [get_clocks CLK1] [get_ports out1]
```

See Also

set_output_delay

Tcl Command Documentation Conventions

Designer Tcl Command Reference

remove_scenario

Tcl command; removes a scenario from the constraint database.

```
remove_scenario <name>
```

Arguments

name

Specifies the name of the scenario to delete.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following command removes the scenario named my_scenario:

```
remove_scenario my_scenario
```

See Also

create_scenario

remove_set

Tcl command; removes a set of paths from analysis. Only user-created sets can be deleted.

```
remove_set -name name
```

Parameters

-name *name*

Specifies the name of the set to delete.

Example

The following command removes the set named my_set:

```
remove_set -name my_set
```

See Also

`create_set`

rename_scenario

Tcl command; renames the specified timing scenario with the new name provided. You must provide a unique new name (that is, it cannot already be used by another timing scenario).

```
rename_scenario oldname -new newname
```

Arguments

oldname

Specifies the current name of the timing scenario.

`-new` *newname*

Specifies the new name to give to the timing scenario.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

This command changes the name of the timing scenario in the list of scenarios.

Example

```
rename_scenario scenario_A -new scenario_B
```

See Also

`create_scenario`

`delete_scenario`

Tcl documentation conventions

Designer Tcl Command Reference

save

Tcl command; saves all changes made prior to this command. This includes changes made on constraints, options and sets.

```
save
```

Arguments

None

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following script sets the maximum number of paths reported by list_paths to 10, reads an SDC file, and save both the option and the constraints into the design project:

```
set_options -limit_max_paths 10
read_sdc somefile.sdc
save
```

See Also

set_options

set_clock_latency

Tcl command; defines the delay between an external clock source and the definition pin of a clock within SmartTime.

```
set_clock_latency -source [-rise][-fall][-early][-late] delay clock
```

Arguments

`-source`

Specifies the source latency on a clock pin, potentially only on certain edges of the clock.

`-rise`

Specifies the edge for which this constraint will apply. If neither or both rise are passed, the same latency is applied to both edges.

`-fall`

Specifies the edge for which this constraint will apply. If neither or both rise are passed, the same latency is applied to both edges.

`-invert`

Specifies that the generated clock waveform is inverted with respect to the reference clock.

`-late`

Optional. Specifies that the latency is late bound on the latency. The appropriate bound is used to provide the most pessimistic timing scenario. However, if the value of "-late" is less than the value of "-early", optimistic timing takes place which could result in incorrect analysis. If neither or both "-early" and "-late" are provided, the same latency is used for both bounds, which results in the latency having no effect for single clock domain setup and hold checks.

`-early`

Optional. Specifies that the latency is early bound on the latency. The appropriate bound is used to provide the most pessimistic timing scenario. However, if the value of "-late" is less than the value of "-early", optimistic timing takes place which could result in incorrect analysis. If neither or both "-early" and "-late" are provided, the same latency is used for both bounds, which results in the latency having no effect for single clock domain setup and hold checks.

`delay`

Specifies the latency value for the constraint.

`clock`

Specifies the clock to which the constraint is applied. This clock must be constrained.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Clock source latency defines the delay between an external clock source and the definition pin of a clock within SmartTime. It behaves much like an input delay constraint. You can specify both an "early" delay

and a"late" delay for this latency, providing an uncertainty which SmartTime propagates through its calculations. Rising and falling edges of the same clock can have different latencies. If only one value is provided for the clock source latency, it is taken as the exact latency value, for both rising and falling edges.

Examples

The following example sets an early clock source latency of 0.4 on the rising edge of main_clock. It also sets a clock source latency of 1.2, for both the early and late values of the falling edge of main_clock. The late value for the clock source latency for the falling edge of main_clock remains undefined.

```
set_clock_latency –source –rise –early 0.4 { main_clock  }
set_clock_latency –source –fall    1.2 { main_clock }
```

See Also

create_clock

create_generated_clock

Tcl Command Documentation Conventions

Designer Tcl Command Reference

set_clock_to_output

SDC command; defines the timing budget available inside the FPGA for an output relative to a clock.

```
set_clock_to_output delay_value -clock clock_ref [–max] [–min] [–clock_fall] output_list
```

Arguments

delay_value

Specifies the clock to output delay in nanoseconds. This time represents the amount of time available inside the FPGA between the active clock edge and the data change at the output port.

`-clock` *clock_ref*

Specifies the reference clock to which the specified clock to output is related. This is a mandatory argument.

`-max`

Specifies that *delay_value* refers to the maximum clock to output at the specified output. If you do not specify –max or –min options, the tool assumes maximum and minimum clock to output delays to be equal.

`-min`

Specifies that *delay_value* refers to the minimum clock to output at the specified output. If you do not specify –max or –min options, the tool assumes maximum and minimum clock to output delays to be equal.

`-clock_fall`

Specifies that the delay is relative to the falling edge of the reference clock. The default is the rising edge.

output_list

Provides a list of output ports in the current design to which delay_value is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

set_clock_uncertainty

Tcl command; specifies a clock-to-clock uncertainty between two clocks (from and to) and returns the ID of the created constraint if the command succeeded.

```
set_clock_uncertainty uncertainty -from | -rise_from | -fall_from from_clock_list -to | -
rise_to | -fall_to to_clock_list -setup {value} -hold {value}
```

Arguments

uncertainty

Specifies the time in nanoseconds that represents the amount of variation between two clock edges.

`-from`

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the source clock list. Only one of the `-from`, `-rise_from`, or `-fall_from` arguments can be specified for the constraint to be valid.

`-rise_from`

Specifies that the clock-to-clock uncertainty applies only to rising edges of the source clock list. Only one of the `-from`, `-rise_from`, or `-fall_from` arguments can be specified for the constraint to be valid.

`-fall_from`

Specifies that the clock-to-clock uncertainty applies only to falling edges of the source clock list. Only one of the `-from`, `-rise_from`, or `-fall_from` arguments can be specified for the constraint to be valid.

from_clock_list

Specifies the list of clock names as the uncertainty source.

`-to`

Specifies that the clock-to-clock uncertainty applies to both rising and falling edges of the destination clock list. Only one of the `-to`, `-rise_to` , or `-fall_to` arguments can be specified for the constraint to be valid.

`-rise_to`

Specifies that the clock-to-clock uncertainty applies only to rising edges of the destination clock list. Only one of the `-to`, `-rise_to` , or `-fall_to` arguments can be specified for the constraint to be valid.

`-fall_to`

Specifies that the clock-to-clock uncertainty applies only to falling edges of the destination clock list. Only one of the `-to`, `-rise_to` , or `-fall_to` arguments can be specified for the constraint to be valid.

to_clock_list

Specifies the list of clock names as the uncertainty destination.

`-setup`

Specifies that the uncertainty applies only to setup checks. If none or both `-setup` and `-hold` are present, the uncertainty applies to both setup and hold checks.

`-hold`

Specifies that the uncertainty applies only to hold checks. If none or both `-setup` and `-hold` are present, the uncertainty applies to both setup and hold checks.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

The set_clock_uncertainty command sets the timing uncertainty between two clock waveforms or maximum clock skew. Timing between clocks have no uncertainty unless you specify it.

Examples

```
set_clock_uncertainty 10 -from Clk1 -to Clk2
set_clock_uncertainty 0 -from Clk1 -fall_to { Clk2 Clk3 } -setup
set_clock_uncertainty 4.3 -fall_from { Clk1 Clk2 } -rise_to *
set_clock_uncertainty 0.1 -rise_from [ get_clocks { Clk1 Clk2 } ] -fall_to { Clk3 Clk4 }
-setup
set_clock_uncertainty 5 -rise_from Clk1 -to [ get_clocks {*} ]
```

See Also

create_clock
create_generated_clock
remove_clock_uncertainty
Designer Tcl Command Reference

set_current_scenario

Tcl command; specifies the timing scenario for the Timing Analyzer to use. All commands that follow this command will apply to the specified timing scenario.

```
set_current_scenario name
```

Arguments

name

Specifies the name of the timing scenario to which to apply all commands from this point on.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

A timing scenario is a set of timing constraints used with a design. If the specified scenario is already the current one, this command has no effect.

After setting the current scenario, constraints can be listed, added, or removed, the checker can be invoked on the set of constraints, and so on.

This command uses the specified timing scenario to compute timing analysis.

Example

```
set_current_scenario scenario_A
```

See Also

get_current_scenario
Tcl Command Documentation Conventions
Designer Tcl Command Reference

set_disable_timing

Tcl command; disables timing arcs within a cell and returns the ID of the created constraint if the command succeeded.

```
set_disable_timing -from value -to value name
```

Arguments

-from *from_port*

Specifies the starting port. The –from and –to arguments must either both be present or both omitted for the constraint to be valid.

-to *to_port*

Specifies the ending port. The –from and –to arguments must either both be present or both omitted for the constraint to be valid.

name

Specifies the cell name where the timing arcs will be disabled.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
set_disable_timing -from A -to Y a2
```

See Also

Tcl documentation conventions

Designer Tcl Command Reference

set_external_check

SDC command; defines the external setup and hold delays for an input relative to a clock.

```
set_external_check delay_value -clock clock_ref [–setup] [–hold] [–clock_fall] input_list
```

Arguments

delay_value

Specifies the external setup or external hold delay in nanoseconds. This time represents the amount of time available inside the FPGA for the specified input after a clock edge.

`-clock` *clock_ref*

Specifies the reference clock to which the specified external check is related. This is a mandatory argument.

`-setup`

Specifies that *delay_value* refers to the setup check at the specified input. This is a mandatory argument if –hold is not used. You must specify either the -setup or -hold option.

`-clock_fall`

Specifies that the delay is relative to the falling edge of the reference clock. The default is the rising edge.

input_list

Provides a list of input ports in the current design to which *delay_value* is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

set_false_path

Tcl command; identifies paths that are considered false and excluded from the timing analysis in the current timing scenario.

```
set_false_path [-from from_list] [-through through_list] [-to to_list]
```

Arguments

`-from` *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through` *through_list*

Specifies a list of pins, ports, cells, or nets through which the disabled paths must pass.

`-to` *to_list*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

The `set_false_path` command identifies specific timing paths as being false. The false timing paths are paths that do not propagate logic level changes. This constraint removes timing requirements on these false paths so that they are not considered during the timing analysis. The path starting points are the input ports or register clock pins, and the path ending points are the register data pins or output ports. This constraint disables setup and hold checking for the specified paths.

The false path information always takes precedence over multiple cycle path information and overrides maximum delay constraints. If more than one object is specified within one -through option, the path can pass through any objects.

You must specify at least one of the `–from`, `-to`, or `–through` arguments for this constraint to be valid.

Examples

The following example specifies all paths from clock pins of the registers in clock domain clk1 to data pins of a specific register in clock domain clk2 as false paths:

```
set_false_path –from [get_clocks {clk1}] –to reg_2:D
```

The following example specifies all paths through the pin U0/U1:Y to be false:

```
set_false_path -through U0/U1:Y
```

See Also

Tcl Command Documentation Conventions

Designer Tcl Command Reference

set_input_delay

Tcl command; creates an input delay on a port list by defining the arrival time of an input relative to a clock in the current scenario.

```
set_input_delay delay_value -clock clock_ref [–max] [–min] [–clock_fall] input_list
```

Arguments

delay_value

Specifies the arrival time in nanoseconds that represents the amount of time for which the signal is available at the specified input after a clock edge.

`-clock` *clock_ref*

Specifies the clock reference to which the specified input delay is related. This is a mandatory argument. If you do not specify -max or -min options, the tool assumes the maximum and minimum input delays to be equal.

`-max`

Specifies that delay_value refers to the longest path arriving at the specified input. If you do not specify -max or -min options, the tool assumes maximum and minimum input delays to be equal.

`-min`

Specifies that delay_value refers to the shortest path arriving at the specified input. If you do not specify -max or -min options, the tool assumes maximum and minimum input delays to be equal.

`-clock_fall`

Specifies that the delay is relative to the falling edge of the clock reference. The default is the rising edge.

input_list

Provides a list of input ports in the current design to which delay_value is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.and IGLOOe, except ProASIC3 nano and ProASIC3L

Description

The set_input_delay command sets input path delays on input ports relative to a clock edge. This usually represents a combinational path delay from the clock pin of a register external to the current design. For in/out (bidirectional) ports, you can specify the path delays for both input and output modes. The tool adds input delay to path delay for paths starting at primary inputs.

A clock is a singleton that represents the name of a defined clock constraint. This can be:

- a single port name used as source for a clock constraint
- a single pin name used as source for a clock constraint; for instance reg1:CLK. This name can be hierarchical (for instance toplevel/block1/reg2:CLK)
- an object accessor that will refer to one clock: [get_clocks {clk}]

Examples

The following example sets an input delay of 1.2ns for port data1 relative to the rising edge of CLK1:

```
set_input_delay 1.2 -clock [get_clocks CLK1] [get_ports data1]
```

The following example sets a different maximum and minimum input delay for port IN1 relative to the falling edge of CLK2:

```
set_input_delay 1.0 -clock_fall -clock CLK2 –min {IN1}
set_input_delay 1.4 -clock_fall -clock CLK2 –max {IN1}
```

See Also

set_output_delay

Tcl Command Documentation Conventions

Designer Tcl Command Reference

set_max_delay

Tcl command; specifies the maximum delay for the timing paths in the current scenario.

```
set_max_delay delay_value [-from from_list] [-to to_list] [-through through_list]
```

Arguments

delay_value

Specifies a floating point number in nanoseconds that represents the required maximum delay value for specified paths.

- If the path starting point is on a sequential device, the tool includes clock skew in the computed delay.
- If the path starting point has an input delay specified, the tool adds that delay value to the path delay.

- If the path ending point is on a sequential device, the tool includes clock skew and library setup time in the computed delay.
- If the ending point has an output delay specified, the tool adds that delay to the path delay.

`-from` *`from_list`*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-to` *`to_list`*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-through` *`through_list`*

Specifies a list of pins, ports, cells, or nets through which the timing paths must pass.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

This command specifies the required maximum delay for timing paths in the current design. The path length for any startpoint in from_list to any endpoint in to_list must be less than delay_value.

The timing engine automatically derives the individual maximum delay targets from clock waveforms and port input or output delays.

The maximum delay constraint is a timing exception. This constraint overrides the default single cycle timing relationship for one or more timing paths. This constraint also overrides a multicycle path constraint.

You must specify at least one of the `–from`, `-to`, or `–through` arguments for this constraint to be valid.

Examples

The following example sets a maximum delay by constraining all paths from ff1a:CLK or ff1b:CLK to ff2e:D with a delay less than 5 ns:

```
set_max_delay 5 -from {ff1a:CLK ff1b:CLK} -to {ff2e:D}
```

The following example sets a maximum delay by constraining all paths to output ports whose names start by “out” with a delay less than 3.8 ns:

```
set_max_delay 3.8 -to [get_ports out*]
```

See Also

`set_min_delay`

`remove_max_delay`

Tcl Command Documentation Conventions

Designer Tcl Command Reference

set_min_delay

Tcl command; specifies the minimum delay for the timing paths in the current scenario.

```
set_min_delay delay_value [-from from_list] [-to to_list] [-through through_list]
```

Arguments

`delay_value`

Specifies a floating point number in nanoseconds that represents the required minimum delay value for specified paths.

- If the path starting point is on a sequential device, the tool includes clock skew in the computed delay.
- If the path starting point has an input delay specified, the tool adds that delay value to the path delay.
- If the path ending point is on a sequential device, the tool includes clock skew and library setup time in the computed delay.
- If the ending point has an output delay specified, the tool adds that delay to the path delay.

`-from` *`from_list`*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-to` *`to_list`*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

`-through` *`through_list`*

Specifies a list of pins, ports, cells, or nets through which the timing paths must pass.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

This command specifies the required minimum delay for timing paths in the current design. The path length for any startpoint in from_list to any endpoint in to_list must be less than delay_value.

The timing engine automatically derives the individual minimum delay targets from clock waveforms and port input or output delays.

The minimum delay constraint is a timing exception. This constraint overrides the default single cycle timing relationship for one or more timing paths. This constraint also overrides a multicycle path constraint.

You must specify at least one of the `–from`, `-to`, or `–through` arguments for this constraint to be valid.

Examples

The following example sets a minimum delay by constraining all paths from ff1a:CLK or ff1b:CLK to ff2e:D with a delay less than 5 ns:

```
set_min_delay 5 -from {ff1a:CLK ff1b:CLK} -to {ff2e:D}
```

The following example sets a minimum delay by constraining all paths to output ports whose names start by “out” with a delay less than 3.8 ns:

```
set_min_delay 3.8 -to [get_ports out*]
```

See Also

`set_max_delay`

`remove_min_delay`

Tcl Command Documentation Conventions

Designer Tcl Command Reference

set_multicycle_path

Tcl command; defines a path that takes multiple clock cycles in the current scenario.

```
set_multicycle_path ncycles [-setup] [-hold] [-from from_list[–through through_list[-to
to_list
```

Arguments

ncycles

Specifies an integer value that represents a number of cycles the data path must have for setup or hold check. The value is relative to the starting point or ending point clock, before data is required at the ending point.

`-setup`

Optional. Applies the cycle value for the setup check only. This option does not affect the hold check. The default hold check will be applied unless you have specified another set_multicycle_path command for the hold value.

`-hold`

Optional. Applies the cycle value for the hold check only. This option does not affect the setup check.

Note: If you do not specify "-setup" or "-hold", the cycle value is applied to the setup check and the default hold check is performed (*ncycles* -1).

`-from` *from_list*

Specifies a list of timing path starting points. A valid timing starting point is a clock, a primary input, an inout port, or a clock pin of a sequential cell.

`-through` *through_list*

Specifies a list of pins or ports through which the multiple cycle paths must pass.

`-to` *to_list*

Specifies a list of timing path ending points. A valid timing ending point is a clock, a primary output, an inout port, or a data pin of a sequential cell.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

Setting multiple cycle paths constraint overrides the single cycle timing relationships between sequential elements by specifying the number of cycles that the data path must have for setup or hold checks. If you change the multiplier, it affects both the setup and hold checks.

False path information always takes precedence over multiple cycle path information. A specific maximum delay constraint overrides a general multiple cycle path constraint.

If you specify more than one object within one -through option, the path passes through any of the objects.

You must specify at least one of the `–from`, `-to`, or `–through` arguments for this constraint to be valid.

Exceptions

Multiple priority management is not supported in Microsemi SoC designs. All multiple cycle path constraints are handled with the same priority.

Examples

The following example sets all paths between reg1 and reg2 to 3 cycles for setup check. Hold check is measured at the previous edge of the clock at reg2.

```
set_multicycle_path 3 -from [get_pins {reg1}] –to [get_pins {reg2}]
```

The following example specifies that four cycles are needed for setup check on all paths starting at the registers in the clock domain ck1. Hold check is further specified with two cycles instead of the three cycles that would have been applied otherwise.

```
set_multicycle_path 4 -setup -from [get_clocks {ck1}]
set_multicycle_path 2 -hold -from [get_clocks {ck1}]
```

See Also

remove_multicycle_path

Tcl Command Documentation Conventions

Designer Tcl Command Reference

set_options (SmartFusion2 and IGLOO2)

SmartTime-specific Tcl command; sets options for timing analysis. Some options will also affect timing-driven place-and-route. The same parameters can be changed in the SmartTime Options dialog box in the SmartTime GUI.

```
set_options
 [-max_opcond value ]
 [-min_opcond value]
 [-interclockdomain_analysis value]
 [-use_bibuf_loopbacks value]
 [-enable_recovery_removal_checks value]
 [-break_at_async value]
 [-filter_when_slack_below value]
 [-filter_when_slack_above value]
 [-remove_slack_filters]
 [-limit_max_paths value]
 [-expand_clock_network value]
 [-expand_parallel_paths value]
 [-analysis_scenario value]
 [-tdpr_scenario value]
 [-reset]
```

Arguments

-max_opcond *value*

Sets the operating condition to use for Maximum Delay Analysis. The following table shows the acceptable values for this argument. Default is *worst*.

Value	Description
worst	Use Worst Case conditions for Maximum Delay Analysis
typical	Use Typical conditions for Maximum Delay Analysis
best	Use Best Case conditions for Maximum Delay Analysis

-min_opcond *value*

Sets the operating condition to use for Minimum Delay Analysis. The following table shows the acceptable values for this argument. Default is *best*.

Value	Description
best	Use Best Case conditions for Minimum Delay Analysis

Value	Description
typical	Use Typical conditions for Minimum Delay Analysis
worst	Use Worst Case conditions for Minimum Delay Analysis

-interclockdomain_analysis *value*

Enables or disables inter-clock domain analysis. Default is *yes*.

Value	Description
yes	Enables inter-clock domain analysis
no	Disables inter-clock domain analysis

-use_bibuf_loopbacks *value*

Instructs the timing analysis whether to consider loopback path in bidirectional buffers (D->Y, E->Y)as false-path {no}. Default is *yes*; i.e., loopback are false paths.

Value	Description
yes	Enables loopback in bibufs
no	Disables loopback in bibufs

-enable_recovery_removal_checks *value*

Enables recovery checks to be included in max-delay analysis and removal checks in min-delay analysis. Default is *yes*.

Value	Description
yes	Enables recovery and removal checks
no	Disables recovery and removal checks

-break_at_async *value*

Specifies whether or not timing analysis is allowed to cross asynchronous pins (clear, reset of sequential elements). Default is *no*.

Value	Description
yes	Enables breaking paths at asynchronous ports
no	Disables breaking paths at asynchronous ports

-filter_when_slack_below *value*

Specifies a minimum slack value for paths reported by list_paths. Not set by default.

-filter_when_slack_above *value*

Specifies a maximum slack value for paths reported by list_paths. Not set by default.

-remove_slack_filters

Removes the slack minimum and maximum set using -filter_when_slack_below and filter_when_slack_above.

-limit_max_paths *value*

Specifies the maximum number of paths reported by list_paths. Default is *100*.

-expand_clock_network *value*

Specify whether or not clock network details are reported in expand_path. Default is *yes*.

Value	Description
yes	Enables expanded clock network information in paths
no	Disables expanded clock network information in paths

-expand_parallel_paths *value*

Specify the number of parallel paths {paths with the same ends} to include in expand_path. Default is *1*.

-analysis_scenario *value*

Specify the constraint scenario to be used for timing analysis. Default is *Primary*, the default scenario.

-tdpr_scenario *value*

Specify the constraint scenario to be used for timing-driven place-and-route. Default is Primary, the default scenario.

-reset

Reset all options to the default values, except those for analysis and TDPR scenarios, which remain unchanged.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Examples

The following script commands the timing engine to use best operating conditions for both max-delay analysis and min-delay analysis:

```
set_options -max_opcond {best} -min_opcond {best}
```

The following script changes the scenario used by timing-driven place-and-route and saves the change in the Libero project for place-and-route tools to see the change.

```
set_options -tdpr_scenario {My_TDPR_Scenario}
```

See Also

save

set_output_delay

Tcl command; defines the output delay of an output relative to a clock in the current scenario.

```
set_output_delay delay_value -clock clock_ref [-max] [-min] [-clock_fall] output_list
```

Arguments

delay_value

Specifies the amount of time before a clock edge for which the signal is required. This represents a combinational path delay to a register outside the current design plus the library setup time (for maximum output delay) or hold time (for minimum output delay).

-clock *clock_ref*

Specifies the clock reference to which the specified output delay is related. This is a mandatory argument. If you do not specify -max or -min options, the tool assumes the maximum and minimum input delays to be equal.

`-max`

Specifies that delay_value refers to the longest path from the specified output. If you do not specify -max or -min options, the tool assumes the maximum and minimum output delays to be equal.

`-min`

Specifies that delay_value refers to the shortest path from the specified output. If you do not specify -max or -min options, the tool assumes the maximum and minimum output delays to be equal.

`-clock_fall`

Specifies that the delay is relative to the falling edge of the clock reference. The default is the rising edge.

`output_list`

Provides a list of output ports in the current design to which delay_value is assigned. If you need to specify more than one object, enclose the objects in braces ({}).

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Description

The set_output_delay command sets output path delays on output ports relative to a clock edge. Output ports have no output delay unless you specify it. For in/out (bidirectional) ports, you can specify the path delays for both input and output modes. The tool adds output delay to path delay for paths ending at primary outputs.

Examples

The following example sets an output delay of 1.2ns for port OUT1 relative to the rising edge of CLK1:

```
set_output_delay 1.2 -clock [get_clocks CLK1] [get_ports OUT1]
```

The following example sets a different maximum and minimum output delay for port OUT1 relative to the falling edge of CLK2:

```
set_output_delay 1.0 -clock_fall -clock CLK2 –min {OUT1}
set_output_delay 1.4 -clock_fall -clock CLK2 –max {OUT1}
```

See Also

`remove_output_delay`

`set_input_delay`

Tcl Command Documentation Conventions

Designer Tcl Command Reference

write_sdc

Tcl command; writes timing constraints into an SDC file. If multiple constraint scenarios are defined, -scenario allows the user to specify which scenario to write. By default, the current scenario is written.

```
write_sdc
-scenario scenario name
-pin_separator (: | /)
file name
```

Arguments

`-scenario` *`scenario name`*

Specify the scenario to write. By default the current scenario is used.

`-pin_separator` *`sep`*

Specify the pin separator used in the SDC file. It can be either ':' or '/'.

`file name`

Specify the SDC file name.

Example

The following script merges two SDC files and writes the result into a third SDC file:

```
read_sdc first.sdc
read_sdc -add second.sdc
write_sdc merged.sdc
```

See Also

read_sdc, set_current_scenario

VERIFYTIMING (SmartFusion2 and IGLOO2)

SmartFusion2, IGLOO2, and RTG4 – Program Manager

Tcl Flow in the Libero SoC for SmartFusion2 and IGLOO2

Use the following commands to manage and build your project in the Libero SoC.

Design Flow in the Project Manager

The Tcl commands below outline the entire design flow. Once you create a project in the Project Manager you can use the commands below to complete every operation from synthesis to generating an HDL netlist. Click any command to go to the command definition.

run_synthesis [-logfile *name*]

run_simulation [-logfile *name*]

check_hdl -file *filename*

check_schematic -file *filename*

create_symbol [-module *module*]

export_io_constraints_from_adb -adb *filename* -output *outputfilename*

generate_ba_files -adb *filename*

generate_hdl_from_schematic [-module *modulename*]

generate_hdl_netlist [-netlist *filename*] [-run_drc *"TRUE | FALSE"*]

rollback_constraints_from_adb -adb *filename* -output *output_filename*

run_designer [-logfile *filename*] [-script "*script to append*"] [-append_commands "*commands to execute*"] [-adb "*new | open | default*"] [-compile "*TRUE | FALSE*"] [-layout "*TRUE | FALSE*"] [-export_ba "*TRUE | FALSE*"]

run_drc [-netlist *file*] [-gen_hdl *"TRUE | FALSE"*]

Manage Profiles in the Project Manager

add_profile -name *profilename* -type "*synthesis | simulation | stimulus | flashpro | physynth | coreconfig*" -tool *profiletool* -location *tool_location* [-args *tool_parameters*] [-batch "*TRUE | FALSE*"]

edit_profile -name *profilename* -type "*synthesis | simulation | stimulus | flashpro | physynth | coreconfig*" -tool *profiletool* -location *tool_location* [-args *tool_parameters*] [-batch "*TRUE | FALSE*"] [-new_name *name*]

export_profiles -file *name* [-export "*predefined | user | all*"]

remove_profile -name *profile_name*

select_profile -name *profile_name*

Linking Files

change_link_source -file *filename* -path *pathname*

create_links [-hdl_source *file*]* [-stimulus *file*]* [-sdc *file*]* [-pin *file*]* [-dcf *file*]* [-gcf *file*]* [-pdc *file*]* [-crt *file*]* [-vcd *file*]*

export_as_link -file *filename* -path *link_path*

unlink -file *file* [-local *local_filename*]

Set Simulation Options in the Project Manager

```
add_modelsim_path -lib library_name [-path library_path] [-remove " "]
```

Set Device in the Project Manager

```
set_device [-family family] [-die die] [-package package]
```

Miscellaneous Operations in the Project Manager

```
project_settings [-hdl "VHDL | VERILOG"] [-auto_update_modelsim_ini "TRUE | FALSE"] [-
auto_update_viewdraw_ini "TRUE | FALSE"] [-block_mode "TRUE | FALSE"] [-
auto_generate_synth_hdl "TRUE | FALSE"] [-auto_run_drc "TRUE | FALSE"] [-
auto_generate_viewdraw_hdl "TRUE | FALSE"] [-auto_file_detection "TRUE | FALSE"]
refresh
set_option [-synth "TRUE | FALSE"] [-module "module"]
remove_core -name core_name
```

configure_tool (SmartFusion2, IGLOO2, and RTG4)

configure_tool is a general-purpose Tcl command to set the parameters for any tool called by Libero for the SmartFusion2, IGLOO2, and RTG4 families. The command requires the name of the tool and one or more parameters in the format *tool_parameter*:*value*. These parameters are separated and passed to the tool to set up its run.

```
configure_tool
-name {<tool_name>} # Each tool_name has its own set of parameters
-params {<parameter>:<value>} # List of parameters and values
tool_name ::= COMPILE | SYNTHESIZE | PLACEROUTE | GENERATEPROGRAMMINGDATA | PROGRAMDEVICE
| PROGRAM_OPTIONS | PROGRAMMER_INFO |IO_PROGRAM_STATE | SPM | FLASH_FREEZE |
PROGRAM_RECOVERY | USER_PROG_DATA
```

Supported tool_names

The following table lists the supported tool_names..

tool_name	Parameter (-params)	Description
COMPILE	See the topic for parameter names and values.	See the topic for description.
SYNTHESIZE	See the topic for parameter names and values.	See the topic for description.
PLACEROUTE	See the topic for parameter names and values.	See the topic for description.
GENERATEPROGRAMMINGDATA	See the topic for parameter names and values.	See the topic for description.
	See the topic for parameter names and values.	See the topic for description.
PROGRAMDEVICE	See the topic for parameter names and values.	See the topic for description.

tool_name	Parameter (-params)	Description
PROGRAM_OPTIONS	See the topic for parameter names and values.	See the topic for description.
PROGRAMMER_INFO	See the topic for parameter names and values.	See the topic for description.
IO_PROGRAMMING_STATE	See the topic for parameter names and values.	See the topic for description.
SPM	See the topic for parameter names and values.	See the topic for description.
FLASH_FREEZE	See the topic for parameter names and values.	See the topic for description.
PROGRAM_RECOVERY	See the topic for parameter names and values.	See the topic for description.
USER_PROG_DATA	See the topic for parameter names and values.	See the topic for description.

See the SmartFusion2, IGLOO2, and RTG4 Tcl for SoC document for the full list of parameters and values.

Example

```
configure_tool -name {COMPILE} \
  -params { DISPLAY_FANOUT_LIMIT:10}\
  -params {MERGE_SDC:true}
configure_tool -name {SYNTHESIZE} -params {LANGUAGE_VHDL_2008:true}
configure_tool -name {PLACEROUTE} -params {PDPR:false} -params \
  {TDPR:true} -{EFFORT_LEVEL:false} –params {INCRPLACEANDROUTE:false}
```

For example, the command:

```
configure_tool \
  -name {COMPILE} -params {DISPLAY_FANOUT_LIMIT:10} \
  -params {MERGE_SDC:true}
```

sets the COMPILE command options DISPLAY_FANOUT_LIMIT to 10 and MERGE_SDC to true.

There are alternative ways to write these commands to fit your coding style. The following three examples all do the same thing.

Method 1 - single line

```
configure_tool -name {COMPILE} -params {DISPLAY_FANOUT_LIMIT:10} -params {MERGE SDC:true}
```

Method 2 - one statement, multiple lines

```
configure_tool \
  -name {COMPILE} \
  -params {DISPLAY_FANOUT_LIMIT:10} \
  -params {MERGE_SDC:true}
```

Method 3 - multiple statements

```
configure_tool -name {COMPILE} -params {DISPLAY_FANOUT_LIMIT:10}
configure_tool -name {COMPILE} -params {MERGE_SDC:true}
```

See Also

Project Manager Tcl Command Reference

Tcl documentation conventions

export_ba_files (SmartFusion2 and IGLOO2)

Tcl command to export the backannotated files. The backannotated files are <design_name>_ba.v (Verilog backannotated netlist) or <design_name>_ba.vhd (VHDL backannotated netlist) and <design_name>_ba.sdf (Standard Delay Format) timing file. These files are passed to the default simulator for postlayout simulation.

```
export_ba_files
-export_dir {absolute path to folder location}
-export_file_name {name of file}
-vhdl {value}
-min_delay {value}
```

Arguments

`-export_dir {absolute path to directory/folder location}`

Folder/directory location.

`-export_file_name {name of file}`

File name to generate the files. If not specified, it takes <design_name> as the default.

`-vhdl {value}`

Generates the <design_name>_ba.v and <design_name>_ba.sdf when set to 0 and <design_name>_ba.vhd and <design_name>_ba.sdf when set to 1. Default is 0.

-min_delay {*value*}

Set to 1 to export enhanced min delays to include your best-case timing results in your Back Annotated file. Default is 0.

Returns

Returns 0 on success, 1 on failure.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
export_ba_files\
  -export_dir {E:\designs\export\sd1}\
  -export_file_name {test}\
  -vhdl 0\
  -min_delay 1
```

export_bitstream_file (SmartFusion2 and IGLOO2)

Configures the parameters for the bitstream to be exported from Libero.

```
 export_bitstream_file [-file_name file_name]
[-format STP | CHAIN_STP | DAT | SPI]
[-master_file 0 | 1]
[-master_file_components SECURITY | FABRIC | ENVM]
```

```
[-encrypted_uek1_file 1 | 0]
[-encrypted_uek1_file_components FABRIC | ENVM ]
[-encrypted_uek2_file 1 | 0]
[-encrypted_uek2_file_components FABRIC | ENVM ]
[-trusted_facility_file 1 | 0]
[-trusted_facility_file_components FABRIC | ENVM ]
[-add_golden_image 1 | 0]
[-golden_image_address golden_image_address]
[-golden_image_design_version golden_image_design_version]
[-add_update_image 1 | 0]
[-update_image_address update_image_address]
[-update_image_design_version update_image_design_version]
[-serialization_stapl_type SINGLE | MULTIPLE]
[-serialization_target_solution Flashpro_3_4_5 | generic_STAPL_player]
```

Arguments

`-file_name` *`file_name`*

File name must start with design name. Bitstream files will be saved in Libero/export folder. Design name will be used as the file name if the parameter is omitted.

`-format` *`STP | CHAIN_STP | DAT | SPI`*

Specifies the bitstream file formats to be exported. Space is used as a delimiter. STP file will be exported if the parameter is omitted.

Security-related options:

Note: One of the bitstream files must be set to “1”. 1 indicates that this particular file type will be imported; 0 indicates that it will not be exported. For example, if `trusted_facility_file` is set to 1, all other file types must be set to 0.

Or, if `trusted_facility_file` is set to 0, a combination of `master_file` and `euk1_file` and `euk2_file` can be set to 1. In this case, `master_file` must be set to 1.

Bitstream encryption with default key (default security):

`-trusted_facility_file` *`1 | 0`*

Specifies the bitstream file to be exported.

`-trusted_facility_file_components` *`FABRIC | ENVM`*

Specifies the components of the design that will be saved to the bitstream file. The value can be any one or both of FABRIC and ENVM.

Custom security options:

`-master_file` *`0 | 1`*

Specifies the bitstream files to be exported. Depends on the selected security.

`-master_file_components` *`SECURITY | FABRIC | ENVM`*

Specifies the components in the design that will be saved to the bitstream file. The value can be any one or any combination of SECURITY, FABRIC, ENVM.

`-encrypted_uek1_file` *`0 | 1`*

`-encrypted_uek1_file_components` *`FABRIC | ENVM`*

Specifies the components of the design that will be saved to uek1 bitstream. The value can be any one or both of FABRIC and ENVM.

`-encrypted_uek2_file` *`0 | 1`*

`-encrypted_uek2_file_components` *`FABRIC | ENVM`*

Specifies the components of the design that will be saved to uek2 bitstream. The value can be any one or both of FABRIC and ENVM.

Bitstream file to be exported and the components of the design that will be saved to the bitstream file are required.

SPI-related options:

`-add_golden_image 1 | 0`

To enable/disable golden SPI image in SPI directory.

`-golden_image_address`

Hexadecimal address for golden image.

`-golden_image_design_version`

Decimal value for golden image design version.

`-add_update_image 1 | 0`

To enable/disable update SPI image.

`-update_image_address`

Hexadecimal value for update image address.

`-update_image_design_version`

Decimal value for update image design version.

SPI-related options are optional.

Serialization options:

`-serialization_stapl_type`

Serialization stapl file type either single or multiple.

`-serialization_target_solution`

Target programming hardware – Flashpro_3_4_5 or generic_STAPL_player.

Serialization options are optional.

Note that a TCL script file exported from Libero will include all command options. You can modify options you need and remove options you do not need.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

Export a bitstream file:

Export bitstream file for design with default security

```
export_bitstream_file \
  -trusted_facility_file 1
  -trusted_facility_file_components {FABRIC}
```

Export bitstream file for design with custom security options

Export bitstream to master file with FABRIC component.

```
export_bitstream_file \
  -master_file 1 \
  -master_file_components {FABRIC }
```

Export bitstreams to master and uek1 encrypted files. Master file to include security and fabric components, uek1 encrypted file to include FABRIC.

```
export_bitstream_file\
  -file_name {iaptcl1}\
  -format {STP}\
  -master_file 1\
  -master_file_components {SECURITY FABRIC}\
  -encrypted_uek1_file 1\
  -encrypted_uek1_file_components {FABRIC}\
  -encrypted_uek2_file 0\
  -encrypted_uek2_file_components { }\
```

Export SPI directory for programming recovery

```
export_bitstream_file \
  -add_golden_image 1 \
  -golden_image_address {1111} \
  -golden_image_design_version {1} \
  -add_update_image 1 \
  -update_image_address {1211} \
  -update_image_design_version {1} \
```

Export bitstream file for design with MSS/serialization clients

```
export_bitstream_file \
  -file_name {mss1} \
  -format {STP} \
  -trusted_facility_file 1 \
  -trusted_facility_file_components {FABRIC ENVM} \
  -serialization_stapl_type {SINGLE} \
  -serialization_target_solution {FLASHPRO_3_4_5}
```

See Also

Project Manager Tcl Command Reference

export_bsdl_file (SmartFusion2 and IGLOO2)

Tcl command to export the BSDL to a specified file. The exported file has a *.bsd file name extension.

```
export_bsdl_file
-file {absolute path and name of BSDL file}
```

Arguments

`-file {absolute path and name of BSDL file}`

Specifies the *.bsd file.

Returns

Returns 0 on success, 1 on failure.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
export_bsdl_file\
  -file {E:/designs/export/sd1.bsd}
```

export_firmware

This Tcl command exports design firmware configuration data, which consists of:

- Component configuration for MSS/HPMS, FDDR and SERDES blocks instantiating your design
- Compatible firmware drivers for your peripherals

It also creates a workspace and project specific to the IDE tool of your choice (SoftConsole, Keil or IAR).

To open your exported firmware projects you must invoke the third-party development tool (SoftConsole, Keil or IAR) outside Libero SoC.

If you make any changes to your design, you must re-export firmware.

```
export_firmware
-export_dir {D:\Designs\software_drivers}
-create_project {0|1}
-software_ide {SoftConsole|Keil|IAR}
```

Arguments

`-export_dir {`*`absolute or relative path of the folder location`*`}`

Specifies the path and name of folder for the exported firmware.

`-create_project {`*`0|1`*`}`

Generates the workspace and project for the specified IDE tool. Default is 0.

`-software_ide {`*`IDE_toolname`*`}`

Specifies one of three IDE tool name: SoftConsole | IAR | Keil.

If you use -create_project parameter and -software_ide paramter at the same time, Libero exports the workspace and project for that Software IDE tool to the *`export_path`*/{*`SoftConsole|Keil|IAR`*} folder.

Returns

Returns 0 on success, 1 on failure.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
export_firmware \
  -export_dir {D:\Designs\software_drivers} \
  -create_project {1} \
  -software_ide {SoftConsole}
```

export_fp_pdc (SmartFusion2 and IGLOO2)

Tcl command to export the Floorplanning Physical Design Constraint (*.pdc) File. The exported file has a *_fp.pdc file name extension.

```
export_fp_pdc
-file {absolute path and name of *_fp.pdc file}
-mode {PDC_PLACE | PDC_FULL_PLACEMENT}
```

Arguments

`-file {`*`absolute path and name of *_fp.pdc file`*`}`

Specifies the *_fp.pdc file.

`-mode {`*`PDC_PLACE | PDC_FULL_PLACEMENT`*`}`

Use PDC_PLACE to export user's floorplanning constraints, for example, fixed logic and regions.

Use PDC_FULL_PLACEMENT to export information about all of the physical design constraints (I/O constraints, I/O Banks, routing constraints, region constraints, global and local clocks).

Returns

Returns 0 on success, 1 on failure.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
export_fp_pdc\
  -file {E:/designs/export/sd1_fp.pdc}\
  -mode {PDC_FULL_PLACEMENT}
```

export_ibis_file (SmartFusion2 and IGLOO2)

Tcl command to export the IBIS (Input/Output Buffer Information Specification) model report. The exported file has a *.ibs file name extension.

```
export_ibis_file
-file {absolute path and name of *.ibs file}
```

Arguments

`-file {absolute path and name of *.ibs file}`

Specifies the IBIS file to export.

Returns

Returns 0 on success, 1 on failure.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
export_ibis_file\
  -file {E:/designs/export/sd1.ibs}
```

export_io_pdc (SmartFusion2 and IGLOO2)

Tcl command to export the I/O constraints Physical Design Constraint (*.pdc) File. The exported file has a *_io.pdc file name extension.

```
export_io_pdc
-file {absolute path and name of *_io.pdc file}
```

Arguments

`-file {absolute path and name of *_io.pdc file}`

Specifies the *_io.pdc file.

Returns

Returns 0 on success, 1 on failure.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
export_io_pdc\
  -file {E:/designs/export/sd1_io.pdc}
```

export_netlist_file (SmartFusion2 and IGLOO2)

Tcl command to export the netlist after the compile state has completed. The netlist can be either Verilog or VHDL. Microsemi recommends exporting the netlist after the compile state has successfully completed.

```
export_netlist_file
-file {absolute path and filename for netlist}
-vhdl {value}
```

Arguments

`-file {absolute path and filename}`

Specifies the path and name of netlist file.

`-vhdl {value}`

Generates the netlist in VHDL (when set to 1) or Verilog (when set to 0). Default is 0 (Verilog netlist).

Returns

Returns 0 on success, 1 on failure.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
export_netlist_files\
  -file {E:/designs/export/sd1/sd1.v}\
  -vhdl 0
```

export_prog_job (SmartFusion2 and IGLOO2)

Configures the parameters for the programming job to be exported.

```
 export_prog_job [-job_file_name job_file_name] -bitstream_file_type TRUSTED_FACILITY | MASTER
| UEK1 | UEK2 -bitstream_file_components SECURITY | FABRIC | ENVM
```

Arguments

`-job_file_name job_file_name`

Job file will be saved in Libero export folder. Name must start with design name.

`-bitstream_file_type TRUSTED_FACILITY | MASTER | UEK1 | UEK2`

Bitstream file to be included in the programming job. Only one bitstream file can be included in any programming job.

`-bitstream_file_components SECURITY | FABRIC | ENVM`

The list of components to be included in the programming job. Components should be delimited by space.

Bitstream_file_components can be any one of SECURITY, FABRIC, ENVM or any combination of them.

Note: The SECURITY option is available in -bitstream_file_components only when file type is MASTER in –bitstream_file_type.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
export_prog_job \
  -job_file_name {sd1} \
  -bitstream_file_type {MASTER} \
  -bitstream_file_components {SECURITY FABRIC ENVM} #any combination of the three allowed
```

export_sdc_file (SmartFusion2 and IGLOO2)

Tcl command to export the SDC (Synopsys Design Constraint) file for timing constraints. The exported file has a *.sdc file name extension.

```
export_sdc_file
-file {absolute path and name of *.sdc file}
```

Arguments

`-file {absolute path and name of *.sdc file}`

Specifies the SDC file to export.

Returns

Returns 0 on success, 1 on failure.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
export_sdc_file \
  -file {E:/designs/export/sd1.sdc}
```

import_component_data

A Libero SoC general-purpose Tcl command to import component data into an existing Libero project. Component refers to MDDR, FDDR and SERDES peripherals in SmartFusion2 devices. Component Data refers to initialization/configuration register values (*init_reg or *init.mem files) of those peripherals. Use this command if and when:

- The synthesized netlist or HDL files in the existing Libero SoC project contains no component (MDDR, FDDR and SERDES) information AND
- You want to add component s (MDDR, FDDR or SERDES) into the existing design.

```
import_component_data
-module root #name of the top_level (root)
-fddr file_path_and_name # has to be FDDR_init.reg or .mem
-mddr file_path_and_name # has to be MDDR_init.reg or .mem
-serdes0 file_path_and_name # has to be SERDESIF_0_init.reg or .mem
-serdes1 file_path_and_name # has to be SERDESIF_1_init.reg or .mem
-serdes2 file_path_and_name # has to be SERDESIF_2_init.reg or .mem
```

```
-serdes3 file_path_and_name # has to be SERDESIF_3_init.reg or .mem
-envm_cfg file_path_and_name
```

Note: The eNVM config file can have any name.

Note: Either *_init.reg (register configuration file) or *.mem files (memory files) can be used. The two cannot be mixed in the same import_component_data command.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

The following is an example of importing components into IGLOO2 designs created with a Libero SoC pre-v11.4 release.

```
import_component_data \
  -module <root> \
  -fddr <file_path>/FDDR_init.mem \
  -mddr <file_path>/MDDR_init.mem \
  -serdes0 <file_path>/SERDESIF_0_init.mem \
  -serdes1 <file_path>/SERDESIF_1_init.mem \
  -serdes2 <file_path>/SERDESIF_2_init.mem \
  -serdes3 <file_path>/SERDESIF_3_init.mem \
  -envm_cfg <user_cfg_filepath>
```

The following is an example of importing components into IGLOO2 designs created with Libero SoC v11.4 or subsequent releases.

```
import_component_data
  -module <root> \
  -fddr <file_path>/FDDR_init.reg \
  -mddr <file_path>/MDDR_init.reg \
  -serdes0 <file_path>/SERDESIF_0_init.reg \
  -serdes1 <file_path>/SERDESIF_1_init.reg \
  -serdes2 <file_path>/SERDESIF_2_init.reg \
  -serdes3 <file_path>/SERDESIF_3_init.reg \
  -envm_cfg <user_cfg_file_path>
```

The following is an example of importing components into SmartFusion2 designs created with a Libero SoC pre- v11.4 release.

```
import_component_data \
  -module <root> \
  -fddr <file_path>/FDDR_init.reg \
  -mddr <file_path>/MDDR_init.reg \
  -serdes0 <file_path>/SERDESIF_0_init.reg \
  -serdes1 <file_path>/SERDESIF_1_init.reg \
  -serdes2 <file_path>/SERDESIF_2_init.reg \
  -serdes3 <file_path>/SERDESIF_3_init.reg \
  -envm_cfg <user_cfg_file_path>
```

The following is an example of importing components into SmartFusion2 designs created with Libero SoC v11.4 or a subsequent release.

```
import_component_data \
  -module <root> \
  -envm_cfg <user_cfg_file_path>
```

Return Value

Returns 0 on success and 1 on failure.

organize_tool_files

Tcl command; used to specify specific constraint files to be passed to and used by a Libero tool.

```
organize_tool_files \
-tool {tool_name}
-params {tool parameters}
-file {<absolute or relative path to constraint file>} \
-module {$design::work} \
-input_type {value} # Specifies type of input file
```

Arguments

`-tool {<tool_name>}`

Specifies the name of the tool files you want to organize.

`-file {<absolute or relative path to constraint file>}`

Specifies the absolute or relative path to the constraint file; there may be multiple -file arguments (see example below).

`-module {<design::work>}`

Module definition, format is <*$design:work*>.

`-input_type {<constraint>}`

Specifies type of input file. Possible values are:constraint | source | simulation | stimulus | unknown

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
organize_tool_files \
-tool {COMPILE} \
-file {D:/Designs/tests/constraint/io/io_0.pdc} \
-file {D:/Designs/tests/constraint/fp/io_1.pdc} \
-file {D:/Designs/tests/constraint/arbiter.sdc} \
-module {sd1::work} \
-input_type {constraint}
```

publish_block (SmartFusion2, IGLOO2, and RTG4)

Configures the block information to be included in the block information (.cxz) file.

Enabled Block Creation must be turned on in Libero SoC to enable the publication of .cxz file. The published .cxz files are imported into the top-level Libero SoC project as part of the bottom-up design methodology.

```
publish_block
-file {absolute or relative path to the *.cxz file}
-publish_placement {1|0}
-publish_routing {1|0}
-publish_region {1|0}
[-vhdl {0|1}]
```

Arguments

`-file {<absolute or relative path to the location of the *.cxz file>}`

`-publish_placement {1 | 0}`

Set to 1 to publish placement info, 0 to not publish. Default is 1.

`-publish_routing {1 | 0}`

Set to 1 to publish routing info, 0 to not publish. Default is 1.

`-publish_region {1 | 0}`

Set to 1 to publish placement region info, 0 to not publish. Default is 1.

`-vhdl {0 | 1}`

Optional. Set to 0 to publish Verilog netlist, 1 to publish VHDL. Default is 0.

Returns

Returns 0 on success, 1 on failure.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
publish_block\
  -file {E:\designs\sd.cxz}\
  -publish_placement {1}\
  -publish_routing {0}\
  -publish_region {1}\
  -vhdl {0} #Optional
```

run_tool (SmartFusion2, IGLOO2, and RTG4)

run_tool starts the specified tool. For tools that support command files, an optional command file can be supplied through the -script parameter.

```
run_tool
-name {<tool_name >} \
-script {<absolute or relative path to script file>}
```

-script is an optional parameter.

```
tool_name ::= SYNTHESIZE | COMPILE | SIM_PRESYNTH | SIM_POSTSYNTH | SIM_POSTLAYOUT |
PLACEROUTE | VERIFYTIMING | VERIFYPOWER | GENERATEPROGRAMMINGFILE | PROGRAMDEVICE |
CONFIGURE_CHAIN | SMARTDEBUG
```

Return

run_tool returns 0 on success and 1 on failure.

Supported tool_names

The following table lists tool_names for run_tool –name {*tool_name*}.

tool_name	Parameter	Description
SYNTHESIZE	-script {*script_file*}	Runs synthesis on your design.
COMPILE	N/A	Runs Compile with default or configured settings.

tool_name	Parameter	Description
SIM_PRESYNTH	N/A	Runs pre-synthesis simulation with your default simulation tool
SIM_POSTSYNTH	N/A	Runs post-synthesis simulation with your default simulation tool.
SIM_POSTLAYOUT	N/A	Runs post-layout simulation with your default simulation tool.
PLACEROUTE	N/A	Runs Layout with default or configured settings.
VERIFYTIMING	-script {*script_file*}	Runs timing analysis with default settings/configured settings in *script_file*.
VERIFYPOWER	-script {*script_file*}	Runs power analysis with default settings/configured settings in *script_file*.
GENERATEPROGRAMMINGFILE	N/A	Generates the bitstream used for programming within Libero.
PROGRAMDEVICE	N/A	Programs your device with configured parameters.
CONFIGURE_CHAIN	-script {*script_file*}	Takes a script that contains FlashPro-specific Tcl commands and passes them to FlashPro Express for execution.
SMARTDEBUG	-script {*script_file*}	Takes a script that contains SmartDebug-specific Tcl commands and passes them to SmartDebug for execution.

`-script {`*absolute or relative path to script file*`}`

Script file location.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Example

```
run_tool \
  -name {COMPILE}
run_tool \
  -name {SYNTHESIZE} -script {./control_synopsys.tcl}
  #control _synopsys.tcl contains the synthesis-specific Tcl commands)
run_tool \
  -name {VERIFYTIMING} \
  -script {./SmartTime.tcl}
  # Script file containing SmartTime-specific Tcl commands
run_tool \
```

```
  -name {VERIFYPOWER} \
  -script {./SmartPower.tcl}
  # Script file containing SmartPower-specific Tcl commands
run_tool \
  -name {SMARTDEBUG}
  -script {./sd_test.tcl}
# Script file containing SmartDebug-specific Tcl commands
```

Note

Where possible, the value of *tool_name* corresponds to the name of the tool in Libero SoC.

Invoking some tools will cause Libero SoC to automatically run some upstream tools in the design flow. For example, if you run COMPILE, Libero runs synthesis first (as with the SYNTHESIZE command) before it runs COMPILE.

See Also

Project Manager Tcl Command Reference

select_libero_design_device

FlashPro-specifc Tcl command. This command selects the Libero design device for the Programming Connectivity and Interface tool within Libero. This command is needed when the tool cannot automatically resolve the Libero design device when there are two or more identical devices that match the Libero design device in the configured JTAG chain.

Syntax

```
select_libero_design_device -name {device_name}
```

Arguments

`-name {device_name}`

Specifies a user-assigned unique device name in the JTAG chain.

Supported Families

SmartFusion2, IGLOO2

Exceptions

None

Example

```
select_libero_design_device –name {M2S050TS (2)}
select_libero_design_device –name {my_design_device}
```

Note

This Tcl command is typically used in a Flashpro Tcl command script file that is passed to the Libero run_tool command.

```
run_tool –name {CONFIGURE_CHAIN} -script {<flashPro_cmd>.tcl}
```

set_live_probe (SmartFusion2 and IGLOO2)

Tcl command; set_live_probe channels A and/or B to the specified probe point(s). At least one probe point must be specified. Only exact probe name is allowed (i.e. no search pattern that may return multiple points).

```
set_live_probe [-deviceName device_name] [–probeA probe_name] [–probeB probe_name]
```

Arguments

`-deviceName` *`device_name`*

Parameter is optional if only one device is available in the current configuration or set for debug (see SmartDebug user guide for details).

`-probeA` *`probe_ name`*

Specifies target probe point for the probe channel A.

-probeB *`probe_ name`*

Specifies target probe point for the probe channel B.

Supported Families

See the Tcl Commands and Supported Families table for the list of families that support this command.

Exceptions

- You must set a debug file
- The array must be programmed and active
- Active probe read or write operation will affect current settings of Live probe since they use same probe circuitry inside the device
- Setting only one Live probe channel affects the other one, so if both channels need to be set, they must be set from the same call to set_live_probe
- Security locks may disable this function
- In order to be available for Live probe, ProbeA and ProbeB I/O's must be reserved for Live probe respectively

Example

Sets the Live probe channel A to the probe point A12 on device sf2.

```
set_live_probe [-deviceName sf2] [–probeA A12]
```

See Also

Project Manager Tcl Command Reference

SmartFusion2, IGLOO2, and RTG4 – Command Tools

COMPILE (SmartFusion2, IGLOO2, and RTG4)

COMPILE is a command tool used in configure_tool and run_tool. Configure_tool allows you to configure the tool's parameters and values prior to executing the tool. Run_tool executes the tool with the configured parameters.

To compile the design in Libero SoC, you must first configure the compile tool with the configure_tool command and then execute the COMPILE command with the run_tool command.

```
configure_tool -name {COMPILE}
-params {name:value}
[-params {name:value}]
run_tool -name {COMPILE}
```

The following tables list the parameter names and values.

configure_tool –name {COMPILE} parameter:value pair

Name	Value	Description
BLOCK_MODE	Boolean {true | false | 1 | 0}	Set to true or 1 when you have blocks in your design and you want to enable the Block mode. Set it to false or 0 if you don't have blocks in your design. Default is false or 0.
ENABLE_DESIGN_SEPARATION	Boolean {true | false | 1 | 0}	Set to true or 1 to enable design separation mode. Design separation mode is for security and safety-critical application and designs where the design's individual subsystems (design blocks) are separate and independent (in terms of physical layout and programming) to meet your design separation requirements. Default is false or 0.
DISPLAY_FANOUT_LIMIT	Integer	Sets the limit of the number of high fanout nets displayed in the compile report. Default is 10.
MERGE_SDC	Boolean {true | false | 1 | 0}	Set to true or 1 if you want to merge your SDC files with existing timing constraints.

Name	Value	Description
		Default is false.
PDC_IMPORT_HARDERROR	Boolean {true | false | 1 | 0}	Set to true or 1 if you want the COMPILE command to abort when errors are found in the physical design constraints. Default is false.
BLOCK_PLACEMENT_CONFLICTS	String {ERROR|KEEP|LOCK|DISCARD}	Instructs the COMPILE engine what to do when the software encouters a placement conflict. When set to: ERROR - Compile errors out if any instance from a Designer block becomes unplaced. This is the default. KEEP - If some instances get unplaced for any reason, the non-conflicting elements remaining are preserved but not locked. Therefore, the placer can move them into another location if necessary.

Name	Value	Description
		LOCK - If some instances get unplaced for any reason, the non-conflicting elements remaining are preserved and locked. DISCARD – Discards any placement from the block, even if there are no conflicts.
BLOCK_ROUTING_CONFLICTS	String {ERROR|KEEP|LOCK|DISCARD}	Instructs the COMPILE engine what to do when the software encounters a routing conflict. When set to: ERROR - Compile errors out if any route in any preserved net from a Designer block is deleted. This is the default. KEEP – If a route is removed from a net for any

Name	Value	Description
		reason, the routing for the non-conflicting nets is kept unlocked. The router can re-route these nets. LOCK – If routing is removed from a net for any reason, the routing for the non-conflicting nets is kept as locked, and the router will not change them. DISCARD - Discards any routing from the block, even if there are no conflicts.
PA4_GB_MAX_RCLKINT_INSERTION	Integer	Specifies the maximum number of global nets that could be demoted to row-globals. Default is 16, Min is 0 and Max is 50.
PA4_GB_MIN_GB_FANOUT_TO_USE_RCLKINT	Integer	Specifies the Minimum fanout of global nets that could be demoted to row-globals. Default is 300. Min is 25 and Max is 5000.
PA4_GB_MAX_FANOUT_DATA_MOVE	Integer	Specifies the Minimum fanout of non-clock nets to be kept on globals. Default is 5000. Min is 300 and Max is

Name	Value	Description
		200,000.
PA4_GB_COUNT	Integer	The number of available global nets is reported. Minimum for all dies is “0”. Default and Maximum values are die-dependent: 005/010 die: Default = Max = 8 025/050/060/090/150 die: Default=Max=16 RT4G075/RT4G150: Default=24, Max=48.
SET_MITIGATION	Boolean {true | false | 1 | 0}	For RTG4 devices only. Controls the mitigation of Single Event Transient (SET) in the FPGA fabric. When enabled, SET filters are turned on globally to help mitigate radiation-induced transients. Default is false.

run_tool –name {COMPILE} Parameter:value pair

Name	Value	Description
NONE		

Supported Families

SmartFusion2, IGLOO2, RTG4

Example

```
configure_tool -name {COMPILE}
  -params {BLOCK_MODE:false}
  -params {BLOCK_PLACEMENT_CONFLICTS:ERROR}
```

```
  -params {BLOCK_ROUTING_CONFLICTS:ERROR}
  -params {DISPLAY_FANOUT_LIMIT:16}
  -params {ENABLE_DESIGN_SEPARATION:true}
  -params {MERGE_SDC:true}
  -params {PA4_GB_MAX_RCLKINT_INSERTION:16}\
  -params {PA4_GB_MIN_GB_FANOUT_TO_USE_RCLKINT:30}\
  -params {PA4_GB_MAX_FANOUT_DATA_MOVE:2000}\
  -params {PA4_GB_COUNT:8}\
  -params {PDC_IMPORT_HARDERROR:true}
run_tool –name {COMPILE} #Takes no parameters
```

Return

`configure_tool -name {COMPILE}`

Returns 0 on success and 1 on failure.

`run_tool -name {COMPILE}`

Returns 0 on success and 1 on failure.

CONFIGURE_CHAIN (SmartFusion2 and IGLOO2)

CONFIGURE_CHAIN is a command tool used in run_tool. The command run_tool -name {CONFIGURE_CHAIN} takes a script file that contains FlashPro-specific Tcl commands and passes them to FlashPro Express for execution.

```
run_tool -name {CONFIGURE_CHAIN} –script {fpro_cmds.tcl}
```

fpro_cmds.tcl is a Tcl script that contains FlashPro-specific Tcl commands to configure JTAG chain. For details on JTAG chain programming Tcl commands, refer to the Tcl commands section in FlashPro's Online Help.

Do not include any FlashPro project-management commands such as open_project, save_project or close_project in this *fpro_cmds.tcl* script file. The run_tool –name {CONFIGURE_CHAIN} command generates these project-management commands for you.

Note:

For a new Libero project without a JTAG chain, executing this command causes Libero to first add the existing design device to the JTAG chain and then execute the commands from the FlashPro script. If, for example, the FlashPro script *fpro_cmds.tcl* contains commands to add four devices, executing the command run_tool –name {CONFIGURE_CHAIN} -script {*fpro_cmds.tcl*} will create a JTAG chain of the Libero design device and the four devices. For existing Libero projects that already have a JTAG chain, the command is executed on the existing JTAG chain.

Supported Families

SmartFusion2, IGLOO2

Example

```
run_tool –name {CONFIGURE_CHAIN} –script {d:/fpro_cmds.tcl}
#Example fpro_cmds.tcl command file for the –script parameter
add_actel_device \
  -file {./sd_prj/sp_g3/designer/impl1/sd1.stp} \
  -name {dev1}
enable_device -name {M2S050TS_5} -enable 0
add_non_actel_device \
  -ir 2 \
  -tck 1.00 \
```

```
  -name {Non-Microsemi Device}
add_non_actel_device \
  -ir 2 \
  -tck 1.00 \
  -name {Non-Microsemi Device (2)}
remove_device -name {Non-Microsemi Device}
set_device_to_highz -name {M2S050TS_5} -highz 1
add_actel_device \
  -device {M2S050TS} \
  -name {M2S050TS (3)}
select_libero_design_device -name {M2S050TS (3)}
```

Return

Returns 0 on success and 1 on failure.

FLASH_FREEZE (SmartFusion2 and IGLOO2)

FLASH_FREEZE is a command tool used in configure_tool. You use the configure_tool -name {FLASH_FREEZE} command to specify:

- The state of the uRAM and LSRAM when the FPGA fabric is in the Flash Freeze state.
- The MSS clock source when the FPGA fabric is in the Flash Freeze state.

```
configure_tool -name {FLASH_FREEZE}
-params {name:value}
-params {name:value}
```

configure_tool –name {FLASH_FREEZE} parameter:value pair

Params_name	<Params_value>	Description
FF_RAM_STATE	Enum {SUSPEND | SLEEP }	Specifies the uRAM and LSRAM state during Flash Freeze. Default is SUSPEND.
FF_MSS_CLOCK	Enum {RCOSC_1MHZ | RCOSC_50MHZ}	Specifies the MSS Clock Source during Flash Freeze. Default is RCOSC_1MHZ.

Supported Families

SmartFusion2, IGLOO2

Example

```
configure_tool -name {FLASH_FREEZE}\
  -params {FF_RAM_STATE:SUSPEND}\
  -params {FF_MSS_CLOCK:RCOSC_1MHZ}
```

Return

Returns 0 on success and 1 on failure.

GENERATEPROGRAMMINGFILE (SmartFusion2 and IGLOO2)

GENERATEPROGRAMMINGFILE is a command tool used in the run_tool command. The run_tool -name {GENERATEPROGRAMMINGFILE} Tcl command generates the Bitstream used for programming.

```
run_tool -name {GENERATEPROGRAMMINGFILE}
```

This command takes no parameters.

Supported Families

SmartFusion2, IGLOO2

Return

Returns 0 on success and 1 on failure.

PROGRAMDEVICE (SmartFusion2 and IGLOO2)

PROGRAMDEVICE is a command tool used in configure_tool and run_tool. Configure_tool allows you to configure the tool's parameters and values prior to executing the tool. Run_tool executes the tool with the configured parameters.

To program the design in Libero SoC, you must first configure the PROGRAMDEVICE tool with configure_tool command and then execute the PROGRAMDEVICE command with the run_tool command.

Use the commands to configure your programming action and the programming procedures associated with the program action.

```
configure_tool -name {PROGRAMDEVICE}
-params {prog_action:params_value}\
[-params {prog_optional_procedures:params_value}]
run_tool -name {PROGRAMDEVICE}
```

configure_tool –name {PROGRAMDEVICE} parameter:value pair

Name	Value	Description
prog_action	String { PROGRAM | VERIFY | ERASE | DEVICE_INFO | READ_IDCODE | ENC_DATA_AUTHENTICATION | VERIFY_DIGEST}	PROGRAM – Programs all selected family features: FPGA Array, targeted eNVM clients and security settings. VERIFY – Verifies all selected family features: FPGA Array, targeted eNVM clients and security settings. ERASE – Erases the selected family features: FPGA Array and security settings. DEVICE_INFO – Displays the IDCODE, the design name, the checksum, and device security settings and programming environment information programmed into the device.READ_IDCODE – Reads the device ID code from the device.

Name	Value	Description
		ENC_DATA_AUTHENTICATION - Encrypted bitstream authentication data. VERIFY_DIGEST – Calculates the digests for the components included in the bitstream and compares them against the programmed values
prog_optional_procedures	Depends on the action from the prog_action parameter.	This parameter is optional. It is only required when the user wants to enable optional procedure.

run_tool –name {PROGRAMDEVICE} Parameter:value pair

Name	Value	Description
None		

Supported Families

SmartFusion2, IGLOO2

Example

```
configure_tool -name {PROGRAMDEVICE} \
  -params {prog_action:PROGRAM} \
  -params {prog_optional_procedures:DO_VERIFY }
configure_tool -name {PROGRAMDEVICE} -params {prog_action:DEVICE_INFO}
run_tool –name {PROGRAMDEVICE} #Takes no parameters
```

Return

configure_tool -name {PROGRAMDEVICE} returns 0 on success and 1 on failure.

run_tool -name {PROGRAMDEVICE} returns 0 on success and 1 on failure.

PROGRAMMING_BITSTREAM_SETTINGS (RTG4 Only)

PROGRAMMING_BITSTREAM_SETTINGS is a command tool used in configure_tool. Configure_tool -name {PROGRAMMING_BITSTREAM_SETTINGS} sets the bitstream settings for your RTG4 devices.

```
configure_tool -name {PROGRAMMING_BITSTREAM_SETTINGS}
-params {name:value}
-params {name:value}
-params {name:value}
```

The following table lists the parameter names and values.

Name	Value	Description
system_controller_suspend_mode	true | false	When set to true, enables System Controller Suspend Mode when TRSTB is

Name	Value	Description
		low during device power up. You can exit System Controller Suspend Mode by driving TRSTB high during device power up.
disable_digest_check	true | false	When set to true, it disables Probe Read/Write when TRSTB is low during device power up. You can enable Probe Read/Write by driving TRSTB high during device power up.
disable_jtag	true | false	When set to true, disables the JTAG interface when TRSTB is low during device power up. You can enable the JTAG interface by driving TRSTB high during device power up.
disable_fabric_erase_write _verify	true | false	When set to true, disables Fabric Erase/Write/Verify when TRSTB is low during device power up. You can enable Fabric Erase/Write/Verify by driving TRSTB high during device power up.
disable_probe_read_write	true | false	When set to true, disables Probe Read/Write when TRSTB is low during device power up. You can enable Probe Read/Write by driving TRSTB high during device power up.
disable_spi	true | false	When set to true, disables the SPI interface when TRSTB is low during device power up. You can enable the SPI interface by driving TRSTB high during device power up.

Supported Families

RTG4

Example

```
configure_tool \
  -name {PROGRAMMING_BITSTREAM_SETTINGS} \
  -params {disable_digest_check:false} \
  -params {disable_fabric_erase_write_verify:false} \
  -params {disable_jtag:true} \
  -params {disable_probe_read_write:false} \
  -params {disable_spi:false} \
  -params {one_time_programmable:false} \
  -params {system_controller_suspend_mode:true}
```

Return

Returns 0 on success and 1 on failure.

PLACEROUTE (SmartFusion2, IGLOO2, and RTG4)

PLACEROUTE is a command tool used in configure_tool and run_tool. Configure_tool allows you to configure the tool's parameters and values prior to executing the tool. Run_tool executes the PLACEROUTE command tool with the configured parameters.

To place and route the design in Libero SoC, you must first configure the PLACEROUTE tool with the configure_tool command and then execute the PLACEROUTE command with the run_tool command.

```
configure_tool -name {PLACEROUTE}
-params {name:value}
-params {name:value}
-params {name:value}
-params {name:value}
run_tool -name {PLACEROUTE}
```

The following tables list the parameter names and values.

configure_tool –name {PLACEROUTE} parameter:value pair

Name	Value	Description
TDPR	Boolean {true | false | 1 | 0}	Set to true or 1 to enable Timing-Driven Place and Route. Default is 1.
PDPR	Boolean {true | false | 1 | 0}	Set to true or 1 to enable Power-Driven Place and Route. Default is false or 0.
EFFORT_LEVEL	Boolean {true | false | 1 | 0}	Set to true or 1 to enable High Effort Layout to optimize design performance. Default is false or 0.
INCRPLACEANDROUTE	Boolean {true | false | 1 | 0}	Set to true or 1 to use previous placement data as the initial placement for the next run. Default is false or 0.
REPAIR_MIN_DELAY	Boolean {true | false | 1 | 0}	Set to 1 to enable Repair Minimum Delay violations for the router when TDPR option is set to true or 1. Default is false.
NUM_MULTI_PASSES	Integer value {"1" through "25"}	Specifies the number of passes to run. The default is 5. Maximum is 25.
START_SEED_INDEX	Integer from "1" to "101"	Indicates the specific index into the array of random seeds which is to be the starting point for the passes. Its value should range from 1 to 100. If not specified, the default behavior is to continue from the last seed index which was used.
MULTI_PASS_LAYOUT	Boolean {true | false | 1 | 0}	Set to true or 1 to enable Multi-Pass Layout Mode for Place and Route. Default is false or 0.
MULTI_PASS_CRITERIA	{SLOWEST_CLOCK" | SPECIFIC_CLOCK | VIOLATIONS | TOTAL_POWER}	Specifies the criteria used to run multi-pass layout: • SLOWEST ClOCK: Use the slowest clock frequency in the design in a given pass as the performance reference for the layout pass. • SPECIFIC_CLOCK: Use a specific clock

Name	Value	Description
		frequency as the performance reference for all layout passes. • VIOLATIONS: Use the pass that best meets the slack or timing-violations constraints. This is the default. • TOTAL POWER:Specifies the best pass to be the one that has the lowest total power (static + dynamic) out of all layout passes.
SPECIFIC_CLOCK	{Name_of_clock}	Applies only when MULTI_PASS_CRITERIA is set to SPECIFIC_CLOCK. It specifies the name of the clock in the design used for Timing Violation Measurement.
DELAY_ANALYSIS	max | min	Used only when MULTI_PASS_CRITERIA is set to “VIOLATIONS”. Specifies the type of timing violations (slacks) to be examined. The default is 'max'. • max: Use timing violations (slacks) obtained from maximum delay analysis • min: Use timing violations (slacks) obtained from minimum delay analysis.
STOP_ON_FIRST_PASS	Boolean {true | false | 1 | 0}	Applies only when MULTI_PASS_CRITERIA is set to “VIOLATIONS”. It stops performing remaining passes if all timing constraints have been met (when there are no negative slacks reported in the timing violations report). Note: The type of timing violations (slacks) used is determined by the 'DELAY_ANALYSIS ' parameter.
SLACK_CRITERIA	{WORST_SLACK | TOTAL_NEGATIVE_SLACK}	Applies only when MULTI_PASS_CRITERIA is set to VIOLATIONS. Specifies how to evaluate the timing violations (slacks). The default is WORST_SLACK. • WORST_SLACK: The largest amount of negative slack (or least amount of positive slack if all constraints are met) for each pass is identified and then the largest value out of all passes will determine the best pass. This is the default. • TOTAL_NEGATIVE_SLACK: The sum of negative slacks from the first 100 paths for each pass in the Timing Violation report is identified. The largest value out of all passes will determine the best pass. If no negative

Name	Value	Description
		slacks exist for a pass, then use the worst slack to evaluate that pass. Note: The type of timing violations (slacks) used is determined by the 'DELAY_ANALYSIS' parameter.

run_tool –name {PLACEROUTE} Parameter:value pair

Name	Value	Description
NONE		

Supported Families

SmartFusion2, IGLOO2 and RTG4

Example

```
configure_tool -name {PLACEROUTE}\
  -params {TDPR:true}\
  -params {PDPR:false}\
  -params {EFFORT_LEVEL:true}\
  -params {INCRPLACEANDROUTE:false}\
run_tool –name {PLACEROUTE} #Takes no parameters
```

Return

`configure_tool -name {PLACEROUTE}`

Returns 0 on success and 1 on failure.

`run_tool -name {PLACEROUTE}`

Returns 0 on success and 1 on failure.

PROGRAM_OPTIONS (SmartFusion2 and IGLOO2)

PROGRAM_OPTIONS is a command tool used in configure_tool. Configure_tool allows you to configure the tool's parameters and values. This Tcl command is the Tcl equivalent of the Configure Bitstream options in the Design Flow window.

```
configure_tool -name {PROGRAM_OPTIONS}
-params {name:value}
[-params {name:value}]
```

The following table lists the parameter names and values.

configure_tool –name {PROGRAM_OPTIONS} parameter:value pair

Name	Value	Description
program_envm	Boolean {true | false | 1 | 0}	Include eNVM component in the programming bitstream ("true" only if eNVM available in the

Name	Value	Description
		design).
program_fabric	Boolean {true | false | 1 | 0}	Include fabric component in the programming bitstream.
program_mode	String {selected feature}	Only “selected_features” is supported.
program_security	Boolean {true | false | 1 | 0}	Include custom security component in the programming bitstream (“true” only if custom security was defined).

Supported Families

SmartFusion2, IGLOO2

Example

```
configure_tool -name {PROGRAM_OPTIONS}\
  -params {program_envm:false}\
  -params {program_fabric:true}\
  -params {program_mode:selected_features}\
  -params {program_security:true}
```

Return

Returns 0 on success and 1 on failure.

PROGRAM_RECOVERY (SmartFusion2 and IGLOO2)

Configures the parameters in the Configure User Programming Data dialog box within Libero.

```
configure_tool -name {PROGRAM_RECOVERY}
-params {enable_auto_update:value}
-params {enable_prog_recovery:value}
-params {spi_clk_freq:value}
-params {spi_data_transfer_mode:value}
```

Arguments

`Enable_auto_update {true|false}`

Enables auto update when set to true and disables it when set to false.

`Enable_prog_recovery {true|false}`

Enables programming recovery when set to true and disables when set to false.

`Spi_clk_freq {1.00|2.08|3.13|4.16|5.00|6.25|8.30|12.50|25.00}`

SPI clock frequency can be set to one of the values specified above.

`Spi_data_transfer_mode {100}`

SPI data transfer mode will set the values for SPS, SPO and SPH in the UI. SPS has a fixed value of 1. The user can change the values of only SPO and SPH to 0 or 1.

Supported Families

SmartFusion2, IGLOO2

Exceptions

None

Example

The following example sets the auto update and programming recovery to be ON. SPI clock frequency is set to 25MHz. SPO and SPH are set to 0.

```
configure_tool -name {PROGRAM_RECOVERY}\
  -params {enable_auto_update:true}\
  -params {enable_prog_recovery:true}\
  -params {spi_clk_freq:25.00}\
  -params {spi_data_transfer_mode:100}
```

PROGRAMMER_INFO (SmartFusion2 and IGLOO2)

PROGRAMMER_INFO is a command tool used in configure_tool. Configure_tool -name {PROGRAMMER_INFO} sets the programmer settings, similar to the way FlashPro commands set the programmer settings. This command supports all five programmers: FlashPro3, FlashPro4, FlashPro5, FlashPro and FlashPro Lite.

```
configure_tool -name {PROGRAMMER_INFO}
-params [{name:value}]
```

The following tables list the parameter names and values.

configure_tool –name {PROGRAMMER_INFO} parameter:value (FlashPro3)

Name	Value	Description
flashpro3_clk_mode	String {free_running_clk | discrete_clocking}	For FlashPro3 Programmer only.
flashpro3_force_freq	String {OFF | ON}	For FlashPro3 Programmer only.
flashpro3_ freq	Integer (Hertz)	For FlashPro3 Programmer only.
flashpro3_vpump	String {ON | OFF}	For Flash For FlashPro3 Programmer only.

configure_tool –name {PROGRAMMER_INFO} Parameter:value (FlashPro4)

Name	Value	Description
flashpro4_clk_mode	String {free_running_clk | discrete_clocking}	For FlashPro4 Programmer only.
flashpro4_force_freq	String {OFF | ON}	For FlashPro4 Programmer only.

Name	Value	Description
flashpro4_ freq	Integer (Hertz)	For FlashPro4 Programmer only.
flashpro4_vpump	String {ON | OFF}	For FlashPro4 Programmer only.

configure_tool –name {PROGRAMMER_INFO} Parameter:value (FlashPro5)

Name	Value	Description
flashpro5_clk_mode	String {free_running_clk | discrete_clocking}	For FlashPro5 Programmer only.
flashpro5_force_freq	String {OFF | ON}	For FlashPro5 Programmer only.
flashpro5_ freq	Integer (Hertz)	For FlashPro5 Programmer only.
flashpro5_vpump	String {ON | OFF}	For FlashPro5 Programmer only.

configure_tool –name {PROGRAMMER_INFO} Parameter:value (FlashPro)

Name	Value	Description
flashpro_drive_trst	String {OFF | ON}	For FlashPro Programmer only.
flashpro_force_freq	String {OFF | ON}	For FlashPro Programmer only.
flashpro_force_vddp	String {ON | OFF}	For FlashPro Programmer only.
flashpro_freq	Integer (Hertz)	For FlashPro Programmer only.
flashpro_vddl	String {ON| OFF}	For FlashPro Programmer only.
flashpro_vddp	String {2.5V | 3.3V}	For FlashPro Programmer only.
flashpro_vpn	String {ON | OFF}	For FlashPro Programmer only.
flashpro_vpp	String {ON | OFF}	For FlashPro Programmer only.

configure_tool –name {PROGRAMMER_INFO} Parameter:value (FlashPro)

Name	Value	Description
flashprolite_drive_trst	String {OFF | ON}	For FlashPro Lite Programmer only.
flashprolite_force_freq	String {OFF | ON}	For FlashPro Lite Programmer only.
flashprolite_freq	Integer (Hertz)	For FlashPro Lite Programmer only.

Name	Value	Description
flashprolite_vpn	String {ON | OFF}	For FlashPro Lite Programmer only.
flashprolite_vpp	String {ON | OFF}	For FlashPro Lite Programmer only.

For a detailed description of the parameters and values, refer to Programmer Settings in the Libero Online Help.

Supported Families

SmartFusion2, IGLOO2

Examples

For FlashPro3 programmer

```
configure_tool –name {PROGRAMMER_INFO}\
  -params {flashpro3_clk_mode:free_running_clk}\
  -params {flashpro3_force_freq:OFF}\
  -params {flashpro3_freq:400000}\
  -params {flashpro3_vpump:ON}
```

For FlashPro4 programmer

```
configure_tool –name {PROGRAMMER_INFO}\
  -params {flashpro4_clk_mode:free_running_clk\}
  -params {flashpro4_force_freq:OFF}\
  -params {flashpro4_freq:400000}\
  -params {flashpro4_vpump:ON}
```

For FlashPro5 programmer

```
configure_tool –name {PROGRAMMER_INFO}\
  -params {flashpro5_clk_mode:free_running_clk}\
  -params {flashpro5_force_freq:OFF}\
  -params {flashpro5_freq:400000}\
  -params {flashpro5_vpump:ON}
```

For FlashPro programmer

```
configure_tool –name {PROGRAMMER_INFO}\
  -params {flashpro_drive_trst:OFF}\
  -params {flashpro_force_freq:ON}\
  -params {flashpro_force_vddp:ON}\
  -params {flashpro_freq:400000}\
  -params {flashpro_vddl:ON}\
  -params {flashpro_vddp:2.5}\
  -params {flashpro_vpn:ON}\
  -params {flashpro_vpp:ON}
```

For FlashPro Lite Programmer

```
configure_tool –name {PROGRAMMER_INFO}\
  -params {flashprolite_drive_trst:OFF}\
  -params {flashprolite_force_freq:ON}\
  -params {flashprolite_freq:400000}\
```

```
-params {flashprolite_vpn:ON}\
-params {flashprolite_vpp:ON}
```

Return

Returns 0 on success and 1 on failure.

SYNTHESIZE

SYNTHESIZE is a command tool used in configure_tool and run_tool. Configure_tool is a general-purpose Tcl command that allows you to configure a tool's parameters and values prior to executing the tool. The run_tool Tcl command then executes the specified tool with the configured parameters.

To synthesize your design in Libero SoC, you first configure the synthesize tool with the configure_tool command and then execute the command with the run_tool command.

```
configure_tool -name {SYNTHESIZE}
-params {name:value}
run_tool -name {SYNTHESIZE}
```

The following tables list the parameter names and values.

configure_tool –name {SYNTHESIZE} parameter:value pair

Name	Value	Description
LANGUAGE_VERILOG_2001	Boolean {true | false | 1 | 0}	Set to true or 1 when the input HDL language is Verilog 2001.
LANGUAGE_SYSTEM_VLOG	Boolean {true | false | 1 | 0}	Set to true or 1 when the input HDL language is System Verilog.
LANGUAGE_VHDL_2008	Boolean {true | false | 1 | 0}	Set to true or 1 when the input HDL language is VHDL 2008.
CLOCK_GLOBAL	Integer	Specifies the threshold value for Clock pin promotion. The default is 2.
CLOCK_DATA	Integer value between 1000 and 200,000.	Specifies the threshold value for data pin promotion. The default is 5000.
RAM_OPTIMIZED_FOR_POWER	Boolean {true | false | 1 | 0}	Set to true or 1 to optimize RAM for Low Power; RAMS are inferred and configured to ensure the lowest power consumption. Set to false or 0 to optimize RAM for High Speed at the expense of more FPGA resources.
RETIMING	Boolean {true | false |	Set to true or 1 to enable Retiming during synthesis. Set to false or 0 to

Name	Value	Description
	1 | 0}	disable Retiming during synthesis.

Run_tool –name {SYNTHESIZE} Parameter:value pair

Name	Value	Description
NONE		

Supported Families

SmartFusion2, IGLOO2

Example

```
configure_tool -name {SYNTHESIZE}\
  -params {LANGUAGE_VERILOG_2001:true}
run_tool –name {SYNTHESIZE} #Takes no parameters
```

Return

`configure_tool -name {SYNTHESIZE}`

Returns 0 on success and 1 on failure.

`run_tool -name {SYNTHESIZE}`

Returns 0 on success and 1 on failure.

USER_PROG_DATA (SmartFusion2 and IGLOO2)

USER_PROG_DATA is a command tool used in configure_tool. Configure_tool -name {USER_PROG_DATA} sets the Design Version and Silicon Signature in your device.

```
configure_tool -name {USER_PROG_DATA}
-params {name:value}
-params {name:value}
```

The following table lists the parameter names and values.

configure_tool –name {USER_PROG_DATA} parameter:value pair

Name	Value	Description
design_version	Integer (0 through 65535)	Sets the design version. It must be greater than the Back level version in SPM Update Policy.
silicon_signature	Hex {<max length 8 Hex characters>}	32-bit (8 hex characters) silicon signature to be programmed into the device. This field can be read from the device using the JTAG USERCODE instruction.

Supported Families

SmartFusion2, IGLOO2

Example

```
configure_tool -name {USER_PROG_DATA}\
  -params {design_version:255}\
  -params {silicon_signature:abcdffff}
```

Return

Returns 0 on success and 1 on failure.

VERIFYPOWER (SmartFusion2 and IGLOO2)

VERIFYPOWER is a command tool used in run_tool. The command run_tool passes a script file that contains power-specific Tcl commands to the VERIFYPOWER command and executes it.

```
run_tool -name {VERIFYPOWER} –script {power_analysis.tcl}
```

where

<power_analysis.tcl> is a script that contains power-specific Tcl commands. You can include power-specific Tcl commands to generate power reports. See the sample power_analysis Tcl Script below for details.

Return

Returns 0 on success and 1 on failure.

Supported Families

SmartFusion2, IGLOO2

Example

```
run_tool –name {VERIFYPOWER} –script {<power_analysis.tcl>}
```

Sample power_analysis Tcl Script <power_analysis.tcl>

The following example changes SmartPower operating condition settings from the default to 40C junction temperature and 1.25V VDD.

It then creates a report called A4P5000_uSRAM_POWER_64X18_power_report.txt.

Change from pre-defined temperature and voltage mode (COM,IND,MIL) to SmartPower custom

```
smartpower_set_temperature_opcond -use "design"
smartpower_set_voltage_opcond -voltage "VDD" -use "design"
```

Set the custom temperature to 40C ambient temperature.

```
smartpower_temperature_opcond_set_design_wide -typical 40 -best 40 -worst 40
```

Set the custom voltage to 1.25V

```
smartpower_voltage_opcond_set_design_wide -voltage "VDD" –typical 1.25 -best 1.25 -worst
1.25
```

VERIFYTIMING (SmartFusion2 and IGLOO2)

VERIFYTIMING is a command tool used in run_tool. Run_tool passes a script file that contains timing-specific Tcl commands to the VERIFYTIMING command and executes it.

```
run_tool -name {VERIFYTIMING} –script {timing.tcl}
```

where

<timing.tcl> is a script that contains SmartTime-specific Tcl commands. You can include SmartTime-specific Tcl commands to create user path sets and to generate timing reports. See sample the Sample SmartTime Tcl Script below for details.

Supported Families

SmartFusion2, IGLOO2

Example

```
run_tool –name {VERIFYTIMING} –script {<timing.tcl>}
```

Return

Returns 0 on success and 1 on failure.

Sample SmartTime Tcl Script <*timing.tcl*>

```
# Create user path set -from B_reg
create_set -name from_B_reg \
  -source {B_reg*[*]:CLK} \
  -sink {*}
# Create user set -from A, B, C
create_set -name from_in_ports \
  -source {A B C} \
  -sink {*}
# Generate Timing Reports
Report \
  -type timing \
  -analysis min \
  -format text \
  -max_paths 10 \
  -print_paths yes \
  -max_expanded_paths 10 \
  -include_user_sets yes \
  min_timing.rpt
# Export SDC
write_sdc -scenario {Primary} exported.sdc
#save the changes
save
```

Product Support

The Microsemi SoC Products Group backs its products with various support services including a Customer Technical Support Center and Non-Technical Customer Service. This appendix contains information about contacting the SoC Products Group and using these support services.

Contacting the Customer Technical Support Center

Microsemi staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions. The Customer Technical Support Center spends a great deal of time creating application notes and answers to FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support

Microsemi customers can receive technical support on Microsemi SoC products by calling Technical Support Hotline anytime Monday through Friday. Customers also have the option to interactively submit and track cases online at My Cases or submit questions through email anytime during the week.

Web: www.actel.com/mycases

Phone (North America): 1.800.262.1060

Phone (International): +1 650.318.4460

Email: soc_tech@microsemi.com

ITAR Technical Support

Microsemi customers can receive ITAR technical support on Microsemi SoC products by calling ITAR Technical Support Hotline: Monday through Friday, from 9 AM to 6 PM Pacific Time. Customers also have the option to interactively submit and track cases online at My Cases or submit questions through email anytime during the week.

Web: www.actel.com/mycases

Phone (North America): 1.888.988.ITAR

Phone (International): +1 650.318.4900

Email: soc_tech_itar@microsemi.com

Non-Technical Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

Microsemi's customer service representatives are available Monday through Friday, from 8 AM to 5 PM Pacific Time, to answer non-technical questions.

Phone: +1 650.318.2470

.

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA

Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

E-mail: sales.support@microsemi.com

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense and security, aerospace, and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs, and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; security technologies and scalable anti-tamper products; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif. and has approximately 3,400 employees globally. Learn more at **www.microsemi.com**.

5-02-00529-4/07.15