PlanAhead Tcl Command Reference Guide

UG789 (v 13.4) January 18, 2012

XILINX®

XILINX®

Chapter 1

Introduction

Overview of Tcl Capabilities in PlanAhead

The Tool Command Language (Tcl) is the scripting language integrated in the PlanAhead™ tool environment. Tcl is a standard language in the semiconductor industry for design constraints and Synopsys® Design Constraints (SDC).

SDC is the mechanism for communicating timing constraints for FPGA synthesis tools from Synopsys Synplify as well as other vendors, and is a timing constraint industry standard; consequently, the Tcl infrastructure is a "Best Practice" for scripting language.

Tcl lets you perform interactive queries to design tools in addition to executing automated scripts. Tcl offers the ability to "ask" questions interactively of design databases, particularly around tool and design settings and state. Examples are: querying specific timing analysis reporting commands live, applying incremental constraints, and performing queries immediately after to verify expected behavior without re-running any tool steps.

The following sections describe some of the basic capabilities of Tcl with PlanAhead.

Note This chapter is not a comprehensive reference to Tcl commands. This chapter does provide references to Tcl resources, and describes the general capabilities of Tcl in the PlanAhead environment.

Tcl Journal Files

When you invoke the PlanAhead tool, it writes the `PlanAhead.log` file to record the various command and operations performed during the design session. The PlanAhead tool also writes a file called `PlanAhead.jou` which is a journal of just the Tcl commands run during the session that can be used as a source to create new Tcl scripts.

Note A backup version of this file, called `planahead.jou_backup`, is written to save the details of the previous run.

Refer to Appendix A in the *PlanAhead User Guide (UG632)* for information regarding the location of these files.

Tcl Help

The Tcl help command provides information related to the supported Tcl commands.

- help—Returns a list of Tcl command categories.

  ```
  help
  ```

  Command categories are groups of commands performing a specific function, like File I/O for instance.

- `help -category` *`category`*—Returns a list of commands found in the specified category.

  ```
  help -category object
  ```

  This example returns the list of Tcl commands for handling objects.

- `help` *`pattern`*—Returns a list of commands that match the specified search pattern. This form can be used to quickly locate a specific command from a group of commands.

  ```
  help get_*
  ```

  This example returns the list of Tcl commands beginning with get_.

- `help` *`command`*—Provides detailed information related to the specified command.

  ```
  help get_cells
  ```

  This example returns specific information of the get_cells command.

- `help -short` *`command`*—Provides an abbreviated help text for the specified command.

  ```
  help -short get_cells
  ```

Starting the PlanAhead Tool

The PlanAhead tool provides three primary modes of operation:

- The GUI mode (default)
- Starting the PlanAhead tool using a Tcl command-line option (Batch mode)
- Tcl shell mode

The following subsections describe Batch and Tcl shell modes.

Batch Mode

When you start the PlanAhead tool, it looks for a Tcl initialization script in two different locations:

1. *`installdir`*`/planAhead/scripts/init.tcl`
2. *`userdir`*`/Xilinx/PlanAhead/init.tcl`

Where:

installdir is the installation directory where the PlanAhead tool is installed, and

userdir is your home directory.

- For Windows: `%APPDATA%/Xilinx/PlanAhead/init.tcl`
- For Linux: `$HOME/.Xilinx/PlanAhead/init.tcl`

If `init.tcl` exists, in one or both of those locations, the PlanAhead tool sources this file; first from the installation directory and second from your home directory.

- The `init.tcl` file in the installation directory allows a company or design group to support a common initialization script for all users. Anyone starting the PlanAhead tool from that installation location sources that `init.tcl` script.
- The `init.tcl` file in the home directory allows each user to specify additional commands, or to override commands from the tool installation to meet their specific design requirements.

The `init.tcll` file is a standard Tcl command file that can contain any valid Tcl command supported by the PlanAhead tool. You can also source another Tcl script file from within `init.tcl` by adding the following statement:

`source` *`path_to_file/file_name`*`.tcl`

General Tcl Syntax Guidelines

Tcl uses the Linux file separator (/) convention regardless of the OS on which you are operating.

The following subsections describe the general syntax guidelines for using Tcl in the PlanAhead application.

Sourcing a Tcl Script

You can source a Tcl script from a command-line option:

`source` *`file_name`*

Within the PlanAhead GUI you can source a Tcl script from **Tools > Run Tcl Script**.

General Syntax Structure

The general structure of the PlanAhead application Tcl commands is:

```
command [optional_parameters] required_parameters
```

Command syntax is of the verb-noun and verb-adjective-noun structure separated by the underscore ("_") character.

Commands are grouped together with common prefixes when they are related.

- Commands that query things are generally prefixed with get_.
- Commands that set a value or a parameter are prefixed with set_.
- Commands that generate reports are prefixed with report_.

The commands are exposed in the global namespace. Commands are "flattened," meaning there are no "sub-commands" for a command.

Example Syntax

Following is an example of the return format on the **`get_cells -help`** command:

```
get_cells

Description:
Get a list of cells in the current design

Syntax:
get_cells  [-hsc  arg ] [-hierarchical] [-regexp] [-nocase] [-filter  arg ]
           [-of_objects  args ] [-match_style  arg ] [-quiet] [ patterns ]

Returns:
list of cell objects

Usage:
  Name           Optional  Default  Description
  ---------------------------------------------
  -hsc           yes       /        Hierarchy separator
  -hierarchical  yes                Search level-by-level in current instance
  -regexp        yes                Patterns are full regular expressions
  -nocase        yes                Perform case-insensitive matching (valid
                                    only when -regexp specified)
  -filter        yes                Filter list with expression
  -of_objects    yes                Get cells of these pins or nets
  -match_style   yes       sdc      Style of pattern matching, valid values are
                                    ucf, sdc
  -quiet         yes                Ignore command errors
   patterns      yes       *        Match cell names against patterns

Categories:
SDC, XDC, Object
```

Unknown Commands

Tcl contains a list of built-in commands that are generally supported by the language, PlanAhead-specific commands which are exposed to the Tcl interpreter, and user-defined procedures.

Commands that do not match any of these known commands are sent to the OS for execution in the shell from the exec command. This lets users execute shell commands that might be OS-specific. If there is no shell command, then an error message is issued to indicate that no command was found.

Return Codes

Some Tcl commands are expected to provide a return value, such as a list or collection of objects on which to operate. Other commands perform an action but do not necessarily return a value that can be used directly by the user. Some tools that integrate Tcl interfaces return a 0 or a 1 to indicate success or error conditions when the command is run.

To properly handle errors in Tcl commands or scripts, you should use the Tcl built-in command catch. Generally, the catch command and the presence of numbered info, warning, or error messages should be relied upon to assess issues in Tcl scripted flows.

PlanAhead tool Tcl commands return either TCL_OK or TCL_ERROR upon completion. In addition, the PlanAhead application sets the global variable $ERRORINFO through standard Tcl mechanisms.

To take advantage of the $ERRORINFO variable, use the following line to report the variable after an error occurs in the Tcl console:

```
puts $ERRORINFO
```

This reports specific information to the standard display about the error. For example, the following code example shows a Tcl script (procs.tcl) being sourced, and a user-defined procedure (loads) being run. There are a few transcript messages, and then an error is encountered at line 5.

```
Line 1: PlanAhead % source procs.tcl
Line 2: PlanAhead% loads
Line 3: Found 180 driving FFs
Line 4: Processing pin a_reg_reg[1]/Q...
Line 5: ERROR: [HD-Tcl 53] Cannot specify '-patterns' with '-of_objects'.
Line 6: PlanAhead% puts $errorInfo
Line 7: ERROR: [HD-Tcl 53] Cannot specify '-patterns' with '-of_objects'. While executing
"get_ports -of objects $pin" (procedure "my_report" line 6) invoked from within procs.tcl
```

You can add **`puts $ERRORINFO`** into catch clauses in your Tcl script files to report the details of an error when it is caught, or use the command interactively in the Tcl console immediately after an error is encountered to get the specific details of the error.

In the example code above, typing the **`puts $ERRORINFO`** command in line 6, reports detailed information about the command and its failure in line 7.

Sourcing a Tcl Script

A Tcl script can be sourced from one of the command-line options or from the GUI using **Tools > Run Tcl Script**. When you invoke a Tcl script from the GUI, a progress bar is displayed and all operations in the GUI are blocked until the scripts completes.

Because there is no way to interrupt the script while it is running, standard operating system methods of killing a process must be used to force interruption of the tool if necessary. If you kill the process, you lose any work done since your last save.

First Class Tcl Objects and Relationships

The Tcl commands in the PlanAhead application provide direct access to the object models for netlist, devices, and projects. These objects are first-class which means they are more than just a string representation, and they can be operated on and queried. There are a few exceptions to this rule, but generally “things” can be queried as objects, and these objects have properties that can be queried and they have relationships that allow you to get to other objects.

Object Types and Definitions

There are many object types in the PlanAhead application; this chapter provides definitions and explanations of the basic types. The most basic and important object types are associated with entities in a design netlist, and these types are listed in the following subsections:

Cell

A cell is an instance, either primitive or hierarchical inside a netlist. Examples of cells include flip-flops, LUTs, I/O buffers, RAM and DSPs, as well as hierarchical instances which are wrappers for other collections of cells.

Pin

A pin is a point of logical connectivity on a cell. A pin allows the internals of a cell to be abstracted away and simplified for easier use, and can either be on hierarchical or primitive cells. Examples of pins include clock, data, reset, and output pins of a flop.

Port

A port is a special type of hierarchical pin, a pin on the top level netlist object, module or entity. Ports are normally attached to I/O pads and connect externally to the FPGA device.

Net

A net is a wire or collection of wires that eventually be physically connected directly together. Nets can be hierarchical or flat, but always sorts a collection of pins together.

Clock

A clock is a periodic signal that propagates to sequential logic within a design. Clocks can be primary clock domains or generated by clock primitives such as a DCM, PLL, or MMCM. A clock is the rough equivalent to a TIMESPEC PERIOD constraint in UCF and forms the basis of static timing analysis algorithms.

Querying Objects

All first class objects can be queried by a get_ Tcl command that generally has the following syntax:

```
get_object_type pattern
```

Where pattern is a search pattern, which includes if applicable a hierarchy separator to get a fully qualified name. Objects are generally queried by a string pattern match applied at each level of the hierarchy, and the search pattern also supports wildcard style search patterns to make it easier to find objects, for example:

```
get_cells */inst_1
```

This command searches for a cell named inst_1 within the first level of hierarchy under the top-level of hierarchy. To recursively search for a pattern at every level of hierarchy, use the following syntax:

```
get_cells –hierarchical inst_1
```

This command searches every level of hierarchy for any instances that match inst_1.

For complete coverage of syntax, see the specific online help for the individual command:

- `help get_cells`
- `get_cells -help`

Object Properties

Objects have properties that can be queried. Property names are unique for any given object type. To query a specific property for an object, the following command is provided:

```
get_property property_nameobject
```

An example would be the lib_cell property on cell objects, which tells you what UniSim component a given instance is mapped to:

```
get_property lib_cell [get_cell inst_1]
```

To discover all of the available properties for a given object type, use the **report_property** command:

```
report_property [get_cells inst_1]
```

The following table shows the properties returned for a specific object.

Reported Properties for Specified Object

Key	Value	Type
bel	OLOGICE1.OUTFF	string
class	cell	string
iob	TRUE	string
is_blackbox	0	bool
is_fixed	0	bool
is_partition	0	bool
is_primitive	1	bool
is_reconfigurable	0	bool
is_sequential	1	bool
lib_cell	FD	string
loc	OLOGIC_X1Y27	string
name	error	string
primitive_group	FD_LD	string
primitive_subgroup	flop	string
site	OLOGIC_X1Y27	string
type	FD & LD	string
XSTLIB	1	bool

Some properties are read-only and some are user-settable. Properties that map to attributes that can be annotated in UCF or in HDL are generally user-settable through Tcl with the set_property command:

```
set_property loc OLOGIC_X1Y27 [get_cell inst_1]
```

Filtering Based on Properties

The object query get_* commands have a common option to filter the query based on any property value attached to the object. This is a powerful capability for the object query commands. For example, to query all cells of primitive type FD do the following:

```
get_cells * -hierarchical –filter “lib_cell == FD”
```

To do more elaborate string filtering, utilize the =~ operator to do string pattern matching. For example, to query all flip-flop types in the design, do the following:

```
get_cells * -hierarchical –filter “lib_cell =~ FD*”
```

Multiple filter properties can be combined with other property filters with logical OR (||) and AND (&&) operators to make very powerful searches. To query every cell in the design that if of any flop type and has a placed location constraint:

```
get_cells * -hierarchical –filter {lib_cell =~ FD* && loc != “”}
```

Note In the example, the filter option value was wrapped with curly braces {} instead of double quotes. This is normal Tcl syntax that prevents command substitution by the interpreter and allows users to pass the empty string (“”) to the loc property.

Large Lists of Objects

Commands that return more than one object generally return a container that looks and behaves like a native Tcl list. This is a feature of the PlanAhead tool in that it allows dramatic optimization of large collections of Tcl objects handling without the need for special iteration commands like the foreach_in_collection command that other tools have implemented. This is handled with the Tcl built-in **foreach** command.

There are a few nuances with respect to large lists, particularly in the log files and the GUI Tcl console. Typically, when you set a Tcl variable to the result of a **get_*** command, the entire list is echoed to the console and to the log file. For large lists, this is truncated when printed to the console and log to prevent memory overloading of the buffers in the tool.

What is echoed is the list printed to the log and console is truncated and the last element appears to be “…” in the log and console, however the actual list in the variable assignment is still correct and the last element is not an error. An example of this is querying a single cell versus every cell in the design, which can be large:

```
get_cells inst_1
inst_1
get_cells * -hierarchical
XST_VCC XST_GND error readIngressFifo wbDataForInputReg fifoSelect_0 fifoSelect_1 fifoSelect_2 fifoSelect_3 ...
%set x [get_cells * -hierarchical]
XST_VCC XST_GND error readIngressFifo wbDataForInputReg fifoSelect_0 fifoSelect_1 fifoSelect_2 fifoSelect_3 ...
%lindex $x end
bftClk_BUFGP/bufg
%llength $x
4454
```

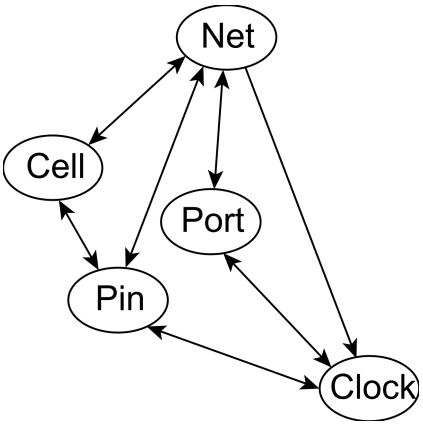
In this example, all four thousand cells were not printed to the console and the list was truncated with a “…” but the actual last element of the list is still correct in the Tcl variable.

Object Relationships

Related objects can be queried using the **-of** option to the relevant **get_*** command. For example, to get a list of pins connected to a cell object, do the following:

```
get_pins –of [get_cells inst_1]
```

The following image shows object types in the PlanAhead tool and their relationships, where an arrow from one object to another object indicates that you can use the **-of** option to the **get_*** command to traverse logical connectivity and get Tcl references to any connected object.



Errors, Warnings, Critical Warnings, and Info Messages

Messages that result from individual commands appear in the log file as well as in the GUI console if it is active. These messages are generally numbered to identify specific issues and are prefixed in the log file with “INFO”, “WARNING”, “CRITICAL_Warning”, “ERROR” followed by a subsystem identifier and a unique number.

The following example shows an INFO message that appears after reading the timing library.

```
INFO: [HD-LIB 1] Done reading timing library
```

These messages make it easier to search for specific issues in the log file to help to understand the context of operations during command execution.

Generally, when an error occurs in a Tcl command sourced from a Tcl script, further execution of subsequent commands is halted. This is to prevent unrecoverable error conditions. There are Tcl built-ins that allow users to intercept these error conditions, and to choose to continue. Consult any Tcl reference for the catch command for a description of how to handle errors using general Tcl mechanisms.

XILINX®

Chapter 2

Tcl Commands Listed by Category

Categories

- COREGenerator
- ChipScope
- FileIO
- Floorplan
- GUIControl
- Object
- PartialReconfiguration
- Partition
- PinPlanning
- Power
- Project
- PropertyAndParameter
- Report
- SDC
- Simulation
- ToolLaunch
- XDC

COREGenerator:

- create_ip
- create_ip_catalog
- generate_ip
- import_ip
- reset_ip

ChipScope:

- connect_debug_port
- create_debug_core
- create_debug_port
- delete_debug_core
- delete_debug_port
- disconnect_debug_port
- get_debug_cores
- get_debug_ports
- implement_debug_core
- launch_chipscope_analyzer
- read_chipscope_cdc
- report_debug_core
- write_chipscope_cdc

FileIO:

- read_chipscope_cdc
- read_csv
- read_edif
- read_pxml
- read_twx
- read_ucf
- read_verilog
- read_vhdl
- read_xdl
- write_bitstream
- write_chipscope_cdc
- write_csv
- write_edif
- write_ibis
- write_ncd
- write_pcf
- write_sdf
- write_timing
- write_ucf
- write_verilog
- write_vhdl
- write_xdc

Floorplan:

- add_cells_to_pblock
- create_pblock
- delete_pblock
- delete_rpm
- get_pblocks
- place_cell
- place_pblocks
- remove_cells_from_pblock
- reset_ucf
- resize_pblock
- swap_locs

GUIControl:

- endgroup
- get_selected_objects
- highlight_objects
- mark_objects
- redo
- select_objects
- start_gui
- startgroup
- stop_gui
- undo
- unhighlight_objects
- unmark_objects
- unselect_objects

Object:

- filter
- get_cells
- get_clocks
- get_debug_cores
- get_debug_ports
- get_designs
- get_files
- get_filesets
- get_generated_clocks
- get_interfaces
- get_iobanks
- get_lib_cells
- get_lib_pins
- get_libs
- get_nets
- get_package_pins
- get_parts
- get_path_groups
- get_pblocks
- get_pins
- get_ports
- get_projects
- get_property
- get_reconfig_modules
- get_runs
- get_selected_objects
- get_sites
- get_timing_arcs
- list_property
- list_property_value
- report_property
- reset_property
- set_property

PartialReconfiguration:

- config_partition
- create_reconfig_module
- delete_reconfig_module
- demote_run
- get_reconfig_modules
- load_reconfig_modules
- promote_run
- reset_property
- set_property
- verify_config

Partition:

- config_partition
- demote_run
- promote_run
- reset_property
- set_property

PinPlanning:

- create_interface
- create_port
- delete_interface
- delete_port
- make_diff_pair_ports
- place_ports
- set_package_pin_val
- split_diff_pair_ports

Power:

- report_power
- reset_switching_activity
- set_switching_activity

Project:

- add_files
- archive_project
- close_design
- close_project
- config_run

- create_fileset
- create_project
- create_run
- current_fileset
- current_project
- current_run
- delete_fileset
- delete_run
- find_top
- get_files
- get_filesets
- get_parts
- get_projects
- get_runs
- help
- import_as_run
- import_files
- import_ip
- import_synplify
- import_xise
- import_xst
- launch_runs
- open_impl_design
- open_io_design
- open_netlist_design
- open_project
- open_rtl_design
- refresh_design
- reimport_files
- remove_files
- reorder_files
- reset_msg_severity
- reset_run
- save_design
- save_design_as
- save_project_as
- set_msg_severity
- set_speed_grade
- update_file
- upgrade_ip
- wait_on_run
- write_bitstream

PropertyAndParameter:

- create_property
- filter
- get_param
- get_property
- list_param
- list_property
- list_property_value
- report_param
- report_property
- reset_param
- reset_property
- set_param
- set_property

Report:

- check_timing
- create_slack_histogram
- delete_clocknetworks_results
- delete_timing_results
- get_msg_count
- get_msg_limit
- report_clock_interaction
- report_clock_utilization
- report_clocknetworks
- report_clocks
- report_config_timing
- report_constraints
- report_control_sets
- report_debug_core
- report_disable_timing
- report_drc
- report_io
- report_min_pulse_width
- report_param
- report_power
- report_property
- report_resources
- report_route_status
- report_ssn
- report_sso
- report_stats
- report_timing
- report_transformed_primitives
- report_utilization
- reset_drc
- reset_msg_limit
- reset_path
- reset_ssn
- reset_sso
- reset_timing
- set_msg_limit
- version

SDC:

- all_clocks
- all_fanin

- all_fanout
- all_inputs
- all_outputs
- all_registers
- create_clock
- create_generated_clock
- current_design
- current_instance
- get_cells
- get_clocks
- get_hierarchy_separator
- get_lib_cells
- get_lib_pins
- get_libs
- get_nets
- get_operating_conditions
- get_pins
- get_ports
- get_timing_arcs
- group_path
- remove_clock
- remove_clock_latency
- remove_data_check
- remove_propagated_clock
- reset_operating_conditions
- reset_timing_derate
- set_case_analysis
- set_clock_gating_check
- set_clock_groups
- set_clock_latency
- set_clock_sense
- set_clock_transition
- set_clock_uncertainty
- set_data_check
- set_disable_timing
- set_false_path
- set_hierarchy_separator
- set_ideal_latency
- set_ideal_network
- set_input_delay
- set_load
- set_logic_dc
- set_logic_one

- set_logic_zero
- set_max_delay
- set_max_fanout
- set_max_time_borrow
- set_min_delay
- set_multicycle_path
- set_operating_conditions
- set_output_delay
- set_propagated_clock
- set_timing_derate
- set_units

Simulation:

- add_files
- compxlib
- create_fileset
- data2mem
- delete_fileset
- import_files
- launch_isim
- remove_files
- write_sdf
- write_verilog
- write_vhdl

ToolLaunch:

- compxlib
- crossprobe_fed
- data2mem
- launch_chipscope_analyzer
- launch_fpga_editor
- launch_impact
- launch_isim
- launch_xpa

XDC:

- all_clocks
- all_cpus
- all_dsps
- all_fanin
- all_fanout

- all_hsios
- all_inputs
- all_mults
- all_outputs
- all_rams
- all_registers
- config_timing_analysis
- config_timing_corners
- config_timing_pessimism
- create_clock
- create_generated_clock
- create_operating_conditions
- create_pblock
- current_design
- current_instance
- delete_power_results
- delete_timing_results
- filter
- get_cells
- get_clocks
- get_default_switching_activity
- get_designs
- get_generated_clocks
- get_hierarchy_separator
- get_interfaces
- get_iobanks
- get_lib_cells
- get_lib_pins
- get_libs
- get_nets
- get_operating_conditions
- get_package_pins
- get_path_groups
- get_pins
- get_ports
- get_property
- get_sites
- get_switching_activity
- get_timing_arcs
- group_path
- remove_clock
- remove_clock_latency
- remove_data_check

- remove_disable_timing
- remove_propagated_clock
- reset_default_switching_activity
- reset_operating_conditions
- reset_property
- reset_switching_activity
- reset_timing_derate
- set_case_analysis
- set_clock_gating_check
- set_clock_groups
- set_clock_latency
- set_clock_sense
- set_clock_transition
- set_clock_uncertainty
- set_data_check
- set_default_switching_activity
- set_delay_model
- set_disable_timing
- set_false_path
- set_hierarchy_separator
- set_ideal_latency
- set_ideal_network
- set_input_delay
- set_input_jitter
- set_load
- set_logic_dc
- set_logic_one
- set_logic_zero
- set_max_delay
- set_max_fanout
- set_max_time_borrow
- set_min_delay
- set_multicycle_path
- set_operating_conditions
- set_output_delay
- set_propagated_clock
- set_property
- set_switching_activity
- set_system_jitter
- set_timing_derate
- set_units

XILINX®

Chapter 3

Tcl Commands Listed Alphabetically

This chapter contains all SDC and Tcl Commands, arranged alphabetically.

add_cells_to_pblock

Add cells to a Pblock

Syntax

```
add_cells_to_pblock [-add_primitives] [-clear_locs]
[-quiet] pblock cells ...
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-add_primitives	yes		Assign all the primitives of the specified instances to a pblock
-clear_locs	yes		Clear instance location constraints
-quiet	yes		Ignore command errors
pblock	no		Pblock to add cells to
cells	no		Cells to add

Categories

Floorplan

Description

This command adds specified logic instances to a Pblock.

Once cells have been added to a Pblock, you can place the Pblocks onto the fabric of the FPGA using the **place_pblocks** command. Once Pblocks have been automatically placed you can manually move pblocks and resize them using the **resize_pblock** command.

You can remove instances from the Pblock using the **remove_cells_from_pblock** command.

Arguments

-add_primitives - Assign all the primitives of the specified instances to a Pblock. This allows you to specify a block module and automatically assign all of the instances within that module to the specified Pblock.

-clear_locs - Clear instance location constraints for any cells that are already placed. This allows you to reset the LOC constraint for cells when defining new Pblocks for floorplanning purposes. When this option is not specified, any instances with assigned placement will not be unplaced as they are added to the Pblock.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

pblock - Specify the name of the Pblock to to add the specified instances to.

cells - Specify one or more cell objects to add to the specified Pblock.

Example

The following example creates a Pblock called pb_cpuEngine, and then adds all of the primitives found in the cpuEngine module, clearing placement constraints for placed instances:

```
create_pblock pb_cpuEngine
add_cells_to_pblock pb_cpuEngine [get_cells cpuEngine] -add_primitives -clear_locs
```

See Also

- get_pblocks
- place_pblocks
- remove_cells_from_pblock
- resize_pblock

add_files

Add sources to the active fileset

Syntax

```
add_files [-fileset arg] [-norecurse] [-scan_for_includes]
[-quiet] [files ...]
```

Returns

list of file objects that were added

Usage

Name	Optional	Default	Description
-fileset	yes		Fileset name
-norecurse	yes		Recursively search in specified directories
-scan_for_includes	yes		Scan and add any included files found in the fileset's RTL sources
-quiet	yes		Ignore command errors
files	yes		Name of the files and/or directories to add. Must be specified if -scan_for_includes is not used.

Categories

Project, Simulation

Description

This command adds one or more source files or the source file contents of one or more directories to the specified fileset.

This command is different from the import_files command, which copies the file into the local project folders as well as adding them to the specified fileset. This command only adds them by reference to the specified fileset.

Arguments

-fileset *name* - Indicates which fileset PlanAhead should add the specified source files to. If the specified fileset does not exist, the PlanAhead tool will return an error. If no fileset is specified the files will be added to the source fileset by default.

files - Provides a list of one or more file names or directory names to be added to the specified fileset. If a directory name is specified the PlanAhead tool will add all valid source files found in the directory, and in subdirectories of the directory.

Note: If the path is not specified as part of the file name, the PlanAhead tool will search for the specified file in the current working directory, or the directory from which the PlanAhead tool was launched.

-norecurse - This argument tells the PlanAhead tool not to recurse through subdirectories of any specified directories. As a default, without this argument specified, the PlanAhead tool will also search through any subdirectories for additional source files that can be added to a project.

-scan_for_includes - Indicates that Verilog source files should be scanned for any '**include** statements and these referenced files should also be added to the specified fileset. As a default the PlanAhead tool will not add '**include** files.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example adds a file called rtl.v to the current project:

```
add_files rtl.v
```

In this example the PlanAhead tool will look for the rtl.v file in the current working directory since no file path is specified, and the file will be added to the source fileset as a default since no fileset is specified.

The following example adds a file called **top.ucf** to the **constrs_1** constraint fileset, as well as any appropriate source files found in the project_1 directory, and its subdirectories:

```
add_files -fileset constrs_1 -quiet c:/Design/top.ucf c:/Design/project_1
```

In the preceding example the PlanAhead tool will locate the **top.ucf** file in the `C:/Design` directory as specified, as well as any constraint files found in the **project_1** directory and its subdirectories, and the files will be added to the specified fileset, in this case the **constrs_1** constraint set.

In addition, the PlanAhead tool will ignore any command line errors because the **-quiet** argument was specified.

If the **-norecurse** option had been specified then only constraint files found in the **project_1** directory would have been added, but subdirectories would not be searched.

See Also

import_files

all_clocks

Get a list of all clocks in the current design

Syntax

```
all_clocks [-quiet]
```

Returns

list of clock objects

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors

Categories

SDC, XDC

Description

Returns a list of all the clocks that have been declared in the current design. To get a list of specific clocks in the design, use the **get_clocks** command.

Clocks can be defined by using the **create_clock** or **create_generated_clock** commands.

Arguments

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example shows all clocks in the sample CPU netlist project:

```
% all_clocks
cpuClk wbClk usbClk phy_clk_pad_0_i phy_clk_pad_1_i fftClk
```

The following example shows how the returned list can be passed to another command:

```
% set_propagated_clock [all_clocks]
```

This will apply the **set_propagated_clock** command to all the clocks in the design.

See Also

- create_clock
- create_generated_clock
- get_clocks
- set_propagated_clock

all_cpus

Get a list of cpu cells in the current design

Syntax

```
all_cpus [-quiet]
```

Returns

list of cpu cell objects

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors

Categories

XDC

Description

Get a list of all CPU cell objects in the current design. Creates a list of all the CPU cell objects that have been declared in the current design.

Note: This command returns a list of CPU cell objects

Arguments

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example returns all CPU objects in the current design:

```
all_cpus
```

The following example shows how the list returned can be passed to another command:

```
set_false_path -from [all_cpus] -to [all_registers]
```

See Also

- all_cpus
- all_dsps
- all_hsios
- all_registers
- set_false_path

all_dsps

Get a list of dsp cells in the current design

Syntax

```
all_dsps [-quiet]
```

Returns

list of dsp cell objects

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors

Categories

XDC

Description

Returns a list of all DSP cell objects in the current design. Creates a list of all the DSP cell objects that have been declared in the current design.

Arguments

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example returns a list of all DSPs defined in the current design:

```
all_dsps
```

The following example shows how the list returned can be passed to another command:

```
set_false_path -from [all_dsps] -to [all_registers]
```

See Also

- all_cpus
- all_hsios
- all_registers
- set_false_path

all_fanin

Get a list of pins or cells in fanin of specified sinks

Syntax

```
 all_fanin [-startpoints_only] [-flat] [-only_cells]
[-levels arg] [-pin_levels arg] [-trace_arcs arg] [-quiet] [to]
```

Returns

list of cell or pin objects

Usage

Name	Optional	Default	Description
-startpoints_only	yes		Find only the timing startpoints
-flat	yes		Hierarchy is ignored
-only_cells	yes		Only cells
-levels	yes	0	Maximum number of cell levels to traverse:Value = 0
-pin_levels	yes	0	Maximum number of pin levels to traverse:Value = 0
-trace_arcs	yes		Type of network arcs to trace: Values: timing, enabled, all
-quiet	yes		Ignore command errors
to	yes		List of sink pins, ports, or nets

Categories

SDC, XDC

Description

This command reports ports, pins or cells in the fan-in of the specified sinks.

Note: This command returns a list of cell, pin, or port objects.

Arguments

-startpoints_only - (Optional) Find only the timing start points.

-flat - (Optional) Ignore the hierarchy of the design.

-only_cells - (Optional) Return only the cell objects which are in the fan-in path of the specified sinks. Do not return pins or ports.

-levels *value* - (Optional) Maximum number of cell levels to traverse. The default value is 0.

-pin_levels *value* - (Optional, Default Value of 0) Maximum number of pin levels to traverse. The default value is 0.

-trace_arcs *value* - Type of network arcs to trace. Valid values are "timing", "enabled", and "all"

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

to - (Optional) Report the fan-in objects to these specified pins, ports, or nets.

Example

The following example lists the timing fan-in of the DAT port:

```
all_fanin DAT
```

See Also

all_fanout

all_fanout

Get a list of pins or cells in fanout of specified sources

Syntax

```
 all_fanout [-endpoints_only] [-flat] [-only_cells]
[-levels arg] [-pin_levels arg] [-trace_arcs arg] [-quiet]
[from]
```

Returns

list of cell or pin objects

Usage

Name	Optional	Default	Description
-endpoints_only	yes		Find only the timing endpoints
-flat	yes		Hierarchy is ignored
-only_cells	yes		Only cells
-levels	yes	0	Maximum number of cell levels to traverse:Value = 0
-pin_levels	yes	0	Maximum number of pin levels to traverse:Value = 0
-trace_arcs	yes		Type of network arcs to trace: Values: timing, enabled, all
-quiet	yes		Ignore command errors
from	yes		List of source pins, ports, or nets

Categories

SDC, XDC

Description

This command reports ports, pins or cells in the fanout of the specified sources.

Note: This command returns a list of cell or pin objects.

Arguments

-endpoints_only - (Optional) Find only the timing endpoints.

-flat - (Optional) Ignore the hierarchy of the design.

-only_cells - (Optional) Return only the cell objects in the fanout path of the specified sources.

-levels *value* - (Optional) Maximum number of cell levels to traverse. The default value is 0.

-pin_levels *value* - (Optional) Maximum number of pin levels to traverse. The default value is 0.

-trace_arcs *value* - Type of network arcs to trace. Valid values are "timing", "enabled", and "all"

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

from - (Optional) List the objects in the fanout path of these specified source ports, pins, or nets.

Example

The following example list the timing fanout of port DAT in the current design:

```
all_fanout -from DAT
```

See Also

all_fanin

all_hsios

Get a list of hsio cells in the current design

Syntax

```
all_hsios [-quiet]
```

Returns

list of hsio cell objects

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors

Categories

XDC

Description

Returns a list of all High Speed IO (HSIO) cell objects that have been declared in the current design. These HSIO cell objects can be assigned to a variable or passed into another command.

Arguments

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example returns all HSIO objects in the current design:

```
all_hsios
```

The following example shows how the list returned can be passed to another command:

```
set_false_path -from [all_hsios] -to [all_registers]
```

See Also

- all_cpus
- all_dsps
- all_registers
- set_false_path

all_inputs

Get a list of all input ports in the current design

Syntax

```
all_inputs [-quiet]
```

Returns

list of port objects

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors

Categories

SDC, XDC

Description

Returns a list of all input port objects in the current design.

Arguments

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example returns all input ports in the current design:

```
all_inputs
```

The following example shows how the list returned can be passed to another command:

```
set_input_delay 5 -clock REFCLK [all_inputs]
```

See Also

- all_outputs
- set_input_delay

all_mults

Get a list of mult cells in the current design

Syntax

```
all_mults [-quiet]
```

Returns

list of mult cell objects

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors

Categories

XDC

Description

Returns a list of all multiplier (MULT) cell objects that have been declared in the current design. These MULT cell objects can be assigned to a variable or passed into another command.

Arguments

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example returns a list of all MULTs in the current design:

```
all_mults
```

The following example shows how the list returned can be passed to another command:

```
set_false_path -from [all_mults] -to [all_registers]
```

See Also

- all_cpus
- all_dsps
- all_hsios
- all_registers
- set_false_path

all_outputs

Get a list of all output ports in the current design

Syntax

```
all_outputs [-quiet]
```

Returns

list of port objects

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors

Categories

SDC, XDC

Description

Returns a list of all output port objects that have been declared in the current design.

Arguments

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example returns all the output ports in the current design:

```
all_outputs
```

The following example sets the output delay for all outputs in the design:

```
set_output_delay 5 -clock REFCLK [all_outputs]
```

See Also

- all_inputs
- set_output_delay

all_rams

Get a list of ram cells in the current design

Syntax

```
all_rams [-quiet]
```

Returns

list of ram cell objects

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors

Categories

XDC

Description

Returns a list of all the RAM cell objects that have been declared in the current design. These RAM cell objects can be assigned to a variable or passed into another command.

Arguments

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example returns all RAM objects in the current design:

```
all_rams
```

See Also

- all_cpus
- all_dsps
- all_hsios
- all_registers

all_registers

Get a list of register cells or pins in the current design

Syntax

```
all_registers [-clock args] [-rise_clock args]
[-fall_clock args] [-cells] [-data_pins] [-clock_pins]
[-async_pins] [-output_pins] [-level_sensitive] [-edge_triggered]
[-quiet]
```

Returns

list of cell or pin objects

Usage

Name	Optional	Default	Description
-clock	yes		Consider registers of this clock
-rise_clock	yes		Consider registers triggered by clock rising edge
-fall_clock	yes		Consider registers triggered by clock falling edge
-cells	yes		Return list of cells (default)
-data_pins	yes		Return list of register data pins
-clock_pins	yes		Return list of register clock pins
-async_pins	yes		Return list of async preset/clear pins
-output_pins	yes		Return list of register output pins
-level_sensitive	yes		Only consider level-sensitive latches
-edge_triggered	yes		Only consider edge-triggered flip-flops
-quiet	yes		Ignore command errors

Categories

SDC, XDC

Description

This command returns a collection of sequential cells or pins in the current design. This command returns a list of all register cells or register pins in the design.

The list of returned objects can be limited by the use of the various arguments described below. You can limit the list of registers returned to a specific clock or clocks, or to registers triggered by the rising or falling edge of a specified clock.

You can also return the pins of collected registers instead of the register objects by specifying one or more of the pin arguments.

Arguments

-cells - (Optional) Returns a list of register cell objects as opposed to a list of pin objects. This is the default behavior of the command.

-clock *args* - (Optional) Returns the list of all registers whose clock pins are in the fanout of the specified clock.

- **-rise_clock** *args* - (Optional) Returns the list of registers triggered by the rising edge of the specified clocks.
- **-fall_clock** *args* - (Optional) Returns the list of registers triggered by the falling edge of the specified clocks.
- Note: The clocks should be specified using either **-clock**, **-rise_clock**, or **-fall_clock**. These arguments should not be combined.

-level_sensitive - (Optional) This option returns the level-sensitive registers or latches.

-edge_triggered - (Optional) This option returns the edge-triggered registers or flip-flops.

-data_pins - (Optional) This option returns a list of data pins of all registers in the design, or of the registers that meet the search requirement.

-clock_pins - (Optional) This option returns a collection of clock pins of the registers that meet the search requirement.

-async_pins - (Optional) This option limits the search to asynchronous pins of the registers that meet the search requirement.

-output_pins - (Optional) This option returns a collection of output pins of the registers that meet the search requirement.

Note: The various **-*_pins** arguments should be used separately. If multiple arguments are specified only one will be used by the command in the following order of precedence: **-data_pins**, **-clock_pins**, **-async_pins**, **-output_pins**.

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example returns a list of registers that are triggered by the falling edge of any clock in the design:

```
all_registers -fall_clock [all_clocks]
```

The following example shows how the list returned can be passed to another command:

```
set_min_delay 2.0 -to [all_registers -clock CCLK -data_pins]
```

See Also

- all_clocks
- set_min_delay

archive_project

Archive the current project

Syntax

```
archive_project [-force] [-exclude_run_results] [-quiet] [file]
```

Returns

true

Usage

Name	Optional	Default	Description
-force	yes		Overwrite existing archived file
-exclude_run_results	yes		Exclude run results from the archive
-quiet	yes		Ignore command errors
file	yes		Name of the archive file

Categories

Project

Description

Use this command to create an archive of a PlanAhead project to store as backup, or to encapsulate the design to send to a remote site.

The PlanAhead tool parses the hierarchy of the design, copies the required source files, include files, and remote files from the library directories, copies the constraint files, copies the results of the various synthesis, simulation, and implementation runs, and then creates a ZIP file of the project.

Arguments

-force - Overwrite an existing ZIP file of the same name. If the ZIP file exists, the PlanAhead tool will return an error unless the **-force** argument is specified.

-exclude_run_results - Excludes the results of any synthesis or implementation runs. This command can greatly reduce the size of a project archive.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

file - Specifies the name of the ZIP file to be created by the **archive_project** command. If *file* is not specified, a ZIP file with the same name as the project will be created.

Examples

The following command archives the current project:

```
archive_project
```

Note: The project archive will be named *project_name*.`zip` because no file name is specified.

The following example specifies `project_3` as the current project, and then archives that project into a file called `proj3.zip`:

```
current_project project_3
archive_project -force -exclude_run_results proj3.zip
```

Note: The use of the **-force** argument causes the PlanAhead tool to overwrite the `proj3.zip` file if one exists. The use of the **-exclude_run_results** argument causes the PlanAhead tool to leave any results from synthesis or implementation runs out of the archive. The various runs defined in the project will be included in the archive, but not any of the results.

See Also

current_project

check_timing

Check timing for possible timing problems for design

Syntax

```
 check_timing [-override_defaults args] [-include args]
[-exclude args] [-verbose] [-quiet] [check_list]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-override_defaults	yes		Overrides the checks in the timing_check_defaults variable using this check_list
-include	yes		Add this list of checks to be performed to timing_check_defaults variable
-exclude	yes		Exclude this list of checks to be performed from timing_check_defaults variable
-verbose	yes		More verbose output
-quiet	yes		Ignore command errors
check_list	yes	{unconstrained _endpointsmultiple_clock no_clock no_input_delay loops generated_clocks}	List of checks to be performed; Values: unconstrained_endpoints, multiple_clock, no_clock,no_input_delay, loops, generated_clocks

Categories

Report

Description

This command checks the design elements of ports, pins, and paths, against the current timing constraints. Use this command to identify possible problems with design data and timing constraints before running the **report_timing** command. The **check_timing** command runs a series of default timing checks, and reports a summary of any violations found. To get detailed information about violations, use the **-verbose** option.

The default timing checks are:

- **generated_clocks** - Checks that derived clocks defined by the **create_generated_clock** command, reference primary clocks, defined by the **create_clock** command, rather than referencing another generated clock.
- **loops** - Checks for and warns of combinational feedback loops in the design.
- **multiple_clock** - Warns if multiple clocks reach a register clock pin. If more than one clock signal reaches a register clock pin it is unclear which clock will be used for analysis. In this case, use the **set_case_analysis** command so that only one clock will propagate to the register clock pin.
- **no_clock** - Reports unclocked registers. In this case, no setup or hold checks are performed on data pins related to the register clock pin.
- **no_input_delay** - Reports the input ports without an input delay constraint. Input delays can be assigned using the **set_input_delay** command. Input ports that are unclocked will not be checked for input delays.
- **unconstrained_endpoints** - This warning identifies timing path endpoints that are not constrained. Endpoints at register data pins are constrained by clock assignment using the **create_clock** command. Endpoints at output ports are constrained by output delays using the **set_output_delay** or **set_max_delay** commands.

Arguments

-override_defaults {*args*} - Overrides the default timing checks and specifies the timing constraint checks to be performed. Specify the checks to be performed from the list of checks described above.

-include *args* - Indicates that the specified checks should be run in addition to the current default checks. Specify the checks to be performed from the list of checks described above.

-exclude *args* - Indicates that the specified checks should be excluded from the default checks performed by the **check_timing** command. Specify the checks to be excluded from the list of checks described above.

-verbose - This option tells the PlanAhead tool to provide detailed results returned from any of the specified checks. The detailed information will include the names of ports or paths with potential timing issues instead of just the number of ports or paths identified by the various checks.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

check_list - Specifies the names of the the checks to be run. Specify the checks to be performed from the list of checks described above.

Examples

The following example runs the default timing checks and returns detailed information for any issues found:

```
check_timing -verbose
```

The following example runs **check_timing**, but excludes the specified checks from the default timing checks:

```
check_timing -exclude {loops generated_clocks}
```

See Also

- create_clock
- create_generated_clock
- report_timing
- set_case_analysis
- set_input_delay
- set_max_delay
- set_output_delay

close_design

Close the current design

Syntax

```
close_design [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors

Categories

Project

Description

Closes the currently active design in the PlanAhead tool.

If the design has been modified, you will not be prompted to save the design prior to closing. You will need to run save_design or save_design_as before using the close_design command to save any changes made to the design.

Arguments

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example closes the current design:

```
close_design
```

If multiple designs are open in the PlanAhead tool, the current design can be specified with the **current_design** command prior to using **close_design**.

The following example sets the current design, then closes it:

```
current_design rtl_1
close_design
```

The **rtl_1** design is set as the active design, then the **close_design** command closes it.

See Also

- current_design
- save_design
- save_design_as

close_project

Close current opened project

Syntax

```
close_project [-delete] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-delete	yes		Delete the project from disk also
-quiet	yes		Ignore command errors

Categories

Project

Description

This command closes the current open project in the PlanAhead tool.

Arguments

-delete - Tells the PlanAhead tool to delete the project data from the hard disk after closing the project.

Note: This argument should be used with caution. The PlanAhead tool will not prompt you to confirm your selection if you specify the **-delete** argument with the **close_project** command.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following command closes the active project:

```
close_project
```

The PlanAhead tool will close the currently opened project. If you have multiple projects open, the **close_project** command applies to the current project which can be defined with the **current_project** command.

The following example sets `project_1` as the active project, and then closes the active project and deletes it from the computer hard disk:

```
current_project project_1
close_project -delete
```

Note: The PlanAhead tool will not prompt you to confirm your command prior to deleting the project files from the computer hard drive. Please be sure this is the desired effect before specifying the **-delete** argument.

See Also

current_project

compxlib

Compile simulation libraries

Syntax

```
compxlib [-arch arg] [-cfg] [-cfgopt arg] [-dir arg] [-e arg]
[-exclude_sublib] [-exclude_superseded] [-force] [-more arg]
[-info arg] [-l arg] [-lib arg] [-log arg] [-p arg] [-s arg]
[-source_lib arg] [-verbose] [-w] [-64bit] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-arch	yes	all	Select device architecture
-cfg	yes	compxlib.cfg	Create new configuration file with default settings
-cfgopt	yes		Configuration option in form of simulator:language:library:options
-dir	yes	.	Directory path for saving the compiled results
-e	yes		Specify the existing directory where previously compxlib-compiled libraries are located
-exclude_sublib	yes		Exclude the sub-lib(s) defined in the edk .pao file for compilation (For edk library only)
-exclude_superseded	yes		Exclude the superseded edk lib(s) for compilation (For edk library only)
-force	yes		Overwrite the pre-compiled libraries
-more	yes		Display extended help on the topic
-info	yes		Print Pre-Compiled library information
-l	yes	all	Compile libraries for this language
-lib	yes	all	Select library to compile
-log	yes	compxlib.log	Create your own log file
-p	yes		Use simulator executables from this directory
-s	yes		Compile libraries for this simulator

Name	Optional	Default	Description
-source_lib	yes		If specified, this directory will be searched for the library source files before searching the default path(s) found in environment variable XILINX (for ISE) or XILINX_EDK (for EDK)
-verbose	yes		Print more messages during program execution
-w	yes		Overwrite the pre-compiled libraries
-64bit	yes		Perform the 64-bit compilation
-quiet	yes		Ignore command errors

Categories

ToolLaunch, Simulation

config_partition

Set module variants and states on a given run

Syntax

```
config_partition [-cell arg] [-reconfig_module arg] [-import]
[-implement] [-import_dir arg] [-preservation arg] [-quiet] run
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-cell	yes		Partition instance to configure in the given run. In order to modify top Partition do not specify this option.
-reconfig_module	yes		Reconfigurable Module variant to apply to this instance in this run
-import	yes		Set this instance (or static logic) to 'import' action for this run
-implement	yes		Set this instance (or static logic) to 'implement' action for this run
-import_dir	yes		Directory from which to import this previously implemented module
-preservation	yes	routing	set the preservation level for the Partition. Values: routing, placement, synthesis
-quiet	yes		Ignore command errors
run	no		Run to be modified

Categories

PartialReconfiguration, Partition

config_run

Configure individual program options for a run

Syntax

```
config_run [-quiet] run program option value
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
run	no		Run to modify
program	no		Name of the program to set an option for
option	no		Name of option to set
value	no		Value of the argument to set for the option

Categories

Project

Description

This command configures the various settings and options of a synthesis or implementation run. The run can be created with the use of the **create_run** command, and can be launched with **launch_runs**.

The **config_run** command defines the value of a single option for a specified synthesis or implementation program. You will need multiple **config_run** statements to define all of the various options needed to configure XST for synthesis, or NGDBUILD, MAP, PAR, and TRACE for implementation.

Refer to the *XST User Guide (ug627)* and the *Command-Line Tools User Guide (ug628)* for more information on each of these tools and the various required and optional settings that can be configured.

Arguments

run - Specifies the name of the synthesis or implementation run to be configured.

-program *arg* - Specifies the program name the defined *option* and *value* apply to. Program names are case sensitive. For the ISE tools, the programs are **xst** for synthesis runs, and **ngdbuild**, **map**, **par**, or **trce** for implementation runs.

-option *arg* - Specifies a single command line option for the specified program. Refer to the *XST User Guide (ug627)* for supported synthesis options, and refer to the *Command-Line Tools User Guide (ug628)* for supported implementation options.

-value *arg* - Specify the value for the defined tool option. Refer to the *XST User Guide (ug627)* for legal values for the specified synthesis options, and refer to the *Command-Line Tools User Guide (ug628)* for legal values for the specified implementation options.

-quiet - Execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example configures the impl_1 run, setting options for NGDBUILD, MAP and PAR:

```
config_run impl_1 -program ngdbuild -option -aul -value true
config_run impl_1 -program map -option -pr -value b
config_run impl_1 -program map -option -cm -value speed
config_run impl_1 -program map -option -ignore_keep_hierarchy -value true
config_run impl_1 -program par -option -pl -value high
config_run impl_1 -program par -option -r -value true
config_run impl_1 -program ngdbuild -option -aut -value true
```

See Also

- create_run
- current_run
- launch_runs

config_timing_analysis

Configure timing analysis general settings

Syntax

```
config_timing_analysis [-disable_paths_between_unrelated_ucf_clocks arg]
[-enable_input_delay_default_clock arg] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-disable_paths _between_unrelated _ucf_clocks	yes		Disable timing paths between unrelated UCF clocks: Values: true, false; This option is not supported for SDC constraints
-enable_input_delay _default_clock	yes		Launch SDC unclocked input delays from an internally defined clock: Values: true, false; This option is not supported for UCF constraints
-quiet	yes		Ignore command errors

Categories

XDC

config_timing_corners

Configure single / multi corner timing analysis settings

Syntax

```
config_timing_corners [-corner arg] [-delay_type arg] [-setup]
[-hold] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-corner	yes		Name of the timing corner to be modified : Values: Slow, Fast
-delay_type	yes		Type of path delays to be analysed for specified timing corner: Values: none, max, min, min_max
-setup	yes		Enable timing corner for setup analysis (equivalent to -delay_type max)
-hold	yes		Enable timing corner for hold analysis (equivalent to -delay_type min)
-quiet	yes		Ignore command errors

Categories

XDC

Description

This command configures the Slow and Fast timing corners for single or multi-corner timing analysis.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-corner *value* - (Optional) Specifies the name of the timing corner to be configured. Valid values are "Slow" and "Fast".

Note: The arguments are case sensitive.

-delay_type *value* - (Optional) Specify the type of path delays to be analysed for the specified timing corner. Valid values are "max", "min" and "min_max".

-setup - (Optional) Specifies setup analysis for the specified timing corner. This is the same as **-delay_type max**.

-hold - (Optional) Specifies hold analysis for the timing corner. This is the same as **-delay_type min**.

Note: You can specify both **-setup** and **-hold** which is the same as **-delay_type min_max**.

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example configures the Slow timing corner for both setup and hold analysis:

```
config_timing_corners -corner Slow -setup -hold
config_timing_corners -corner Slow -delay_type min_max
```

Note: The two preceding examples have the same effect.

The following example configures the Fast corner for min delay analysis:

```
config_timing_corners -corner Fast -delay_type min
```

See Also

- config_timing_analysis
- config_timing_pessimism

config_timing_pessimism

Configure timing analysis common node pessimism removal settings

Syntax

```
config_timing_pessimism [-enable] [-disable] [-transition arg]
[-common_node arg] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-enable	yes		Enable common node pessimism removal
-disable	yes		Disable common node pessimism removal
-transition	yes		Remove pessimism for specified transitions; Values: any_transition, same_transition
-common_node	yes		Perform pessimism removal to common node in routing network. Values: on, off
-quiet	yes		Ignore command errors

Categories

XDC

Description

Enables or disables Clock Reconvergence Pessimism Removal (CRPR). CRPR is disabled by default, which can lead to pessimistic timing results including false hold time violations. To avoid this, CRPR must be manually enabled.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-enable - (Optional) Enable clock reconvergence pessimism removal (CRPR).

-disable - (Optional) Disable clock reconvergence pessimism removal (CRPR).

-transition *arg* - (Optional) Remove pessimism for the specified transitions. Valid values are any_transition or same_transition. The default setting is for any_transition.

-threshold *arg* - (Optional) Common node pessimism threshold.

-common_node *[on, off]* - (Optional) Perform CRPR to common node in routing network.

-quiet - (Optional) Execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example enables CRPR:

```
config_timing_pessimism -common_node on -enable
```

See Also

set_clock_latency

connect_debug_port

Connect nets and pins to debug port channels

Syntax

```
connect_debug_port [-channel_start_index arg]
[-quiet] port nets ...
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-channel_start_index	yes		Connect nets starting at channel index
-quiet	yes		Ignore command errors
port	no		Debug port name
nets	no		List of nets or pins

Categories

ChipScope

Description

Connects a signal from the Netlist design to a port on a Chipscope debug core. The signal can either be connected to a specific channel index on the port, or simply connected to an available channel on the port.

If you try to connect too many signals to a port, or there are not enough channels to support the connection, the PlanAhead tool will return an error.

Additional ports can be added to a debug core through the use of the create_debug_port command, and you can increase the available channels on an existing port with the set_property port_width command. See the examples below.

You can disconnect signals from ports using the disconnect_debug_port command.

When the ChipScope debug core has been defined and connected, you can implement the debug core as a block for inclusion in the Netlist Design. Use the **implement_debug_core** command to use CoreGen to implement the core.

Arguments

-channel_start_index *arg* - Specifies the channel index to use for the connection. If more than one signal has been specified, this is the channel index where connections will start to be added. Channel indexes are numbered starting at 0.

Note: If this argument is not specified, the PlanAhead tool will place connections on the first available channel index.

port - Specifies the name of the port to connect signals to. The port must be referenced by the core_name/port_name.

nets - Specifies a list of one or more net names from the Netlist Design to connect to the specified debug port.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example creates a new TRIG port on the myCore debug core, increases the port_width of the port in order to prepare it to receive the number of signals to be connected, then connects the signals to the port starting at the third channel position (index 2).

```
create_debug_port myCore TRIG
set_property port_width 8 [get_debug_ports myCore/TRIG0]
connect_debug_port myCore/TRIG0 [get_nets [list m0_ack_o m0_cyc_i m0_err_o m0_rty_o \ m0_stb_
```

Note: If you specify too many nets to connect to the available channels on the port, PlanAhead will return an error and will not connect the ports.

See Also

- create_debug_port
- disconnect_debug_port
- get_debug_ports
- get_nets
- implement_debug_core
- set_property

create_clock

Create a clock object

Syntax

```
create_clock [-period arg] [-name arg] [-waveform args] [-add]
[-quiet] [objects]
```

Returns

new clock object

Usage

Name	Optional	Default	Description
-period	yes	10.0	Clock period: Value 0
-name	yes		Clock name
-waveform	yes		Clock edge specification
-add	yes		Add to the existing clock in source_objects
-quiet	yes		Ignore command errors
objects	yes		List of clock source ports, pins or nets

Categories

SDC, XDC

Description

Create a clock object with the specified period or waveform. This command defines primary clocks which are used by the timing engine as the delay propagation starting point of any clock edge. The defined clock can be added to the definition of an existing clock, or overwrite the existing clock.

A virtual clock can be created that has no source in the design. A virtual clock can be used as a time reference for setting input and output delays but does not physically exist in the design.

A clock can also be generated from a primary clock (physical or virtual), and derive many of its properties from the master clock. Use the **create_generated_clock** command to derive a clock from a primary clock.

Note: This command returns the name of the clock object that is created.

Arguments

-period *arg* - (Required) Specifies the clock period of the clock object to be created. The value must be greater than zero (>0), and has a default value of 10.0 time units.

Note: The time units are defined by the **set_units** command. The default time units is nanoseconds (ns).

-name *arg* - (Optional) The name of the clock object to be created. If the name is omitted, a system-generated name will be used based on the specified source *objects*. You can also use the **-name** option without source *objects* to create a virtual clock for the design that is not associated with a physical source on the design.

-waveform *arg1 arg2 ...* - (Optional) Specifies the rise and fall edge times of the waveform of the defined clock, in nanoseconds, over one full clock cycle. There must be an even number of edges, representing both the rising and falling edges of the waveform. The first time specified is the first rising transition, and the second time specified is the falling edge.

Note: If you do not specify the waveform, the default waveform is assumed to have a rising edge at time 0 and a falling edge at one half the specified period

-add - (Optional) Use this option to define multiple clocks on the same source for simultaneous analysis with different clock waveforms. You must use **-name** to specify the new clock to add. If you do not specify this option, the **create_clock** command will automatically assign a name and will overwrite any existing clock of the same name.

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

objects - Specifies the ports, pins, or nets which are the source of the specified clock. If you specify a clock on a source object that already has a clock, the new clock will overwrite the original clock unless you also specify the -add option. If no *objects* are specified to attach the clock object to, the clock will be created as a virtual clock in the design.

Note: The first driver pin of a specified net will be used as the source of the clock

Example

The following example creates a physical clock called bftClk and defines the clock period:

```
create_clock -name bftClk -period 5.000 [get_ports bftClk]
```

See Also

- all_clocks
- create_generated_clock
- get_clocks
- set_input_delay
- set_output_delay
- set_propagated_clock
- set_clock_transition
- set_units

create_debug_core

Create a new ChipScope debug core

Syntax

```
create_debug_core [-quiet] name type
```

Returns

new debug_core object

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
name	no		Name of the new debug core instance
type	no		Type of the new debug core

Categories

ChipScope

Description

Defines a new ChipScope debug core to be added to an open Netlist Design in the current project. The debug core defines ports for connecting nets to for debug purposes.

The default core that is created includes a CLK port and a trigger (TRIG) port. The CLK port only supports one clock signal, and so you must create a separate debug core for each clock domain.

Once the core is created you can add new ports to the debug core with the create_debug_port command, and connect signals to the ports using the connect_debug_port command.

Note: A debug core can only be added to an open Netlist Design in the PlanAhead tool.

Arguments

name - Specifies the name of the ChipScope debug core to add to the project.

type - Specifies the of ChipScope debug core to insert. Currently only the chipscope_ila_v1 type is supported in the PlanAhead tool. The ILA debug core simply adds another load onto a connected net without otherwise altering it. Refer to the *ChipScope Pro Software and Cores User Guide (ug029)* for more information on debug core types and purpose.

Note: When the ILA core is added to the project, the PlanAhead tool also adds an ICON controller core as a container for one or more ILA cores. However, you cannot directly add an ICON core to the project.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example opens the Netlist Design, and creates a new ChipScope debug core:

```
open_netlist_design -name netlist_1
create_debug_core myCore chipscope_ila_v1
```

Note: Currently the chipscope_ila_v1 is the only type of core supported by the PlanAhead tool.

The following example creates a new debug core called myCore and returns the properties of the newly created core:

```
report_property [create_debug_core myCore chipscope_ila_v1]
```

The properties of the debug core can be customized by using the set_property command as in the following example:

```
set_property enable_storage_qualification false [get_debug_cores myCore]
```

See Also

- connect_debug_port
- create_debug_port
- delete_debug_core
- get_debug_cores
- implement_debug_core
- read_chipscope_cdc
- report_debug_core
- report_property
- set_property
- write_chipscope_cdc

create_debug_port

Create a new ChipScope debug port

Syntax

```
create_debug_port [-quiet] name type
```

Returns

new debug_port object

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
name	no		Name of the debug core instance
type	no		Type of the new debug port

Categories

ChipScope

Description

Defines a new port to be added to an existing ChipScope debug core. The port provides connection points to a debug core to attach nets from the design for debug purposes.

When a new debug core is created using the create_debug_core command, it includes a CLK and trigger (TRIG) port by default. However, you can also add DATA and trigger_output (TRIG_OUT) ports to the debug core as well as additional TRIG ports.

A port can have one or more connection points to support one or more nets to debug. As a default new ports are defined as having a width of 1, allowing only one net to be attached. You can change the port width of TRIG and DATA ports to support multiple signals using the set_property port_width command (see Examples).

Note: CLK and TRIG_OUT ports can only have a width of 1.

You can connect signals to ports using the connect_debug_port command, and disconnect signals with the disconnect_debug_port command.

Arguments

name - Specifies the name of the ChipScope debug core to add the new port to. The debug core must already exist in the project having been created with create_debug_port or imported with read_chipscope_cdc.

type - Specifies the type of debug port to insert. There are four port types supported: CLK, DATA, TRIG, and TRIG_OUT. Refer to the *ChipScope Pro Software and Cores User Guide* (ug029) for more information on port types and purpose.

Note: Each ILA debug core can have only one CLK, DATA, and TRIG_OUT port. However, you can create multiple trigger (TRIG) ports.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example creates a new ChipScope debug core, and then adds a DATA port to that core:

```
create_debug_core myCore chipscope_ila_v1
create_debug_port myCore DATA
```

The following example creates a new port on the myCore debug core, and then sets the port width to 8, and begins connecting signals to the port:

```
create_debug_port myCore TRIG
set_property PORT_WIDTH 8 [get_debug_ports myCore/TRIG0]
connect_debug_port -channel_start_index 1 myCore/TRIG0 {m1_cyc_i \
        m1_ack_o m1_err_o m1_rty_o}
```

Note: The debug core is referenced by its name, and the debug port is referenced by the core_name/port_name.

See Also

- connect_debug_port
- create_debug_core
- disconnect_debug_port
- read_chipscope_cdc
- set_property

create_fileset

Create a new fileset

Syntax

```
create_fileset [-constrset] [-simset] [-quiet] name
```

Returns

new fileset object

Usage

Name	Optional	Default	Description
-constrset	yes		Create fileset as constraints fileset (default)
-simset	yes		Create fileset as simulation source fileset
-quiet	yes		Ignore command errors
name	no		Name of the fileset to be create

Categories

Project, Simulation

Description

The create_fileset command is used to define a new fileset within the PlanAhead project.

A fileset is a collection of files with a specific function within the project. One or more constraint files is a constraint set (-constrset); one or more simulation test benches is a simulation set (-simset). Only one fileset option can be specified when using the create_fileset command. As a default, the PlanAhead tool will create a constraint fileset if the type is not specified.

The create_fileset command returns the name of the newly created fileset, or will return an error message unless the -quiet command has been specified.

Arguments

-constrset - Creates a constraint set to hold one or more constraint files. This is the default fileset created if neither the **-constrset** or **-simset** argument is specified.

-simset - creates a simulation fileset to hold one or more simulation source files. You can only specify one type of fileset argument, either **-constrset** or **-simset**. The PlanAhead tool will return an error if both are specified.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute

name - Specifies the name of the fileset to be created.

Examples

The following example creates a new constraint file set named constraints2

```
create_fileset -constrset -quiet constraints2
```

Note: With **-quiet** specified, the PlanAhead tool will not return anything if it encounters an error in trying to create the specified fileset.

The following example creates a new simulation fileset named `sim_1`:

```
create_fileset -simset sim_1
```

Files can be added to the newly created fileset using the **add_files** command.

See Also

- add_files
- current_fileset

create_generated_clock

Create a generated clock object

Syntax

```
create_generated_clock [-name arg] [-source args] [-edges args]
[-divide_by arg] [-multiply_by arg] [-combinational]
[-duty_cycle arg] [-invert] [-edge_shift args] [-add]
[-master_clock arg] [-quiet] [objects]
```

Returns

new clock object

Usage

Name	Optional	Default	Description
-name	yes		Generated clock name
-source	yes		Master clock source object pin/port
-edges	yes		Edge Specification
-divide_by	yes	0	Frequency division factor: Value = 1
-multiply_by	yes	0	Frequency multiplication factor: Value = 1
-combinational	yes		Create a divide_by 1 clock through combinational logic
-duty_cycle	yes	0.0	Duty cycle for frequency multiplication: Range: 0.0 to 100.0
-invert	yes		Invert the signal
-edge_shift	yes		Edge shift specification
-add	yes		Add to the existing clock in source_objects
-master_clock	yes		Use this clock if multiple clocks present at master pin
-quiet	yes		Ignore command errors
objects	yes		List of clock source ports, pins, or nets

Categories

SDC, XDC

Description

Create a new clock object from an existing master clock object in the design.

Note: This command returns the name of the clock object that is created.

Arguments

-name *arg* - (Optional) Specifies the name of the generated clock object.

-source *arg* - (Required) Specifies a pin or port of the master clock to derive the generated clock from. The master clock can be a previously defined physical, virtual, or generated clock.

-master_clock *arg* - (Optional) If there are multiple clocks found on the source pin or port, the specified clock is the one to use as the master for the generated clock object.

-divide_by *arg* - (Optional) Divide the period of the master clock by the specified value to establish the period of the generated clock object. The value specified must be >= 1.0.

-multiply_by *arg* - (Optional) Multiply the period of the master clock by the specified value to establish the period of the generated clock object. The value specified must be >= 1.0.

-combinational - (Optional) Define a combinational path to create a "-divide_by 1" generated clock.

-duty_cycle *arg* - (Optional) Specify the duty cycle of the generated clock as a percentage of the new clock period when used with the -multiply_by argument. The value is specified as a percentage from 0.0 to 100.

-invert - (Optional) The waveform of the generated clock object should be inverted from the waveform of the master clock object.

-edges *arg* - (Optional) Specifies the edges of the master clock to define transitions for the generated clock. Specify edges by their numerical order on the master clock starting with edge 1. Enclose multiple edge numbers in braces {}. See the example below for specifying edge numbers.

-edge_shift *arg* - (Optional) Shift the edges of the generated clock by the specified values relative to the master clock. See the example below for specifying edge shift.

-add - (Optional) Add the generated clock object to an existing clock group specified by *objects*.

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

objects - (Optional) Assign the generated clock object to the specified pin, port, or net.

Examples

The following example defines a generated clock that is divided from the master clock found on the specified CLK pin:

```
create_generated_clock -divide_by 2 -source [get_pins CLK]
```

The following example defines a generated clock named CLK1 from the specified source clock, defining the edges of the master clock to transition the generated clock, with edges shifted by the specified amount, and the generated clock object assigned to the specified pin:

```
create_generated_clock -name CLK1 -source CMB/CLKIN -edges {2 3 8} \
-edge_shift {0 -1.0 -2.0}CMB/CLKOUT
```

The waveform pattern of the generated clock is repeated based on the transitions defined by the **-edges** argument.

See Also

- check_timing
- create_clock
- get_generated_clocks
- set_clock_latency
- set_clock_transition
- set_clock_uncertainty
- set_propagated_clock

create_interface

Create a new I/O port interface

Syntax

```
create_interface [-parent arg] [-quiet] name
```

Returns

new interface object

Usage

Name	Optional	Default	Description
-parent	yes		Assign new interface to this parent interface
-quiet	yes		Ignore command errors
name	no		Name for new I/O port interface

Categories

PinPlanning

create_ip

Create an instance of a configurable IP and add it to the fileset

Syntax

```
create_ip [-srcset arg] -module_name arg -vendor arg -library arg -name arg
```

Returns

list of file objects that were added

Usage

Name	Optional	Default	Description
-srcset	yes		Source set name
-module_name	no		Name for the new IP that will be added to the project
-vendor	no		IP Vendor name
-library	no		IP Library name
-name	no		IP Name
-version	no		IP Version
-quiet	yes		Ignore command errors

Categories

COREGenerator

create_ip_catalog

Create an updated version of the IP Catalog and save it to the indicated directory

Syntax

```
create_ip_catalog -dir arg [-repositories args] [-quiet]
```

Returns

The path to the new IP Catalog file

Usage

Name	Optional	Default	Description
-dir	no		Directory in which to write the new IP Catalog
-repositories	yes		List of IP repositories to be included in the new IP Catalog
-quiet	yes		Ignore command errors

Categories

COREGenerator

create_operating_conditions

Create a new set of operating conditions in a library

Syntax

```
create_operating_conditions -name arg [-library arg]
[-process arg] [-temperature arg] [-voltage arg]
[-tree_type arg] [-calc_mode arg] [-airflow arg]
[-rail_voltages args] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-name	no		Operating Condition Name
-library	yes		Library name
-process	yes	0.0	Process multiplier: 0 to 100
-temperature	yes	varies by family	Ambient Temperature (C): -55 to 125
-voltage	yes	0.0	Voltage (V): 0 to 1000
-tree_type	yes	balanced_tree	Tree type
-calc_mode	yes	nominal	Calculation mode: nominal or worst_case
-airflow	yes	varies by family	Airflow (LFM): 0 to 750
-rail_voltages	yes		List of rail voltage 'name value' pairs (supported: vccint, vccaux, vcco33, vcco25, vcco18, vcco15, vcco12)
-quiet	yes		Ignore command errors

Categories

XDC

create_pblock

Create a new Pblock

Syntax

```
create_pblock [-parent arg] [-quiet] name
```

Returns

new pblock object

Usage

Name	Optional	Default	Description
-parent	yes		parent of the new pblock
-quiet	yes		Ignore command errors
name	no		name of the new pblock

Categories

XDC, Floorplan

Description

This command defines a Pblock to allow you to add logic instances for floorplanning purposes. Creating a Pblock results in an AREA_GROUP constraint that is written to the constraint file.

You can add logic elements to the Pblock using the **add_cells_to_pblock** command, and then place the Pblocks onto the fabric of the FPGA using the **place_pblocks** command. Once Pblocks have been automatically placed you can manually move pblocks and resize them using the **resize_pblock** command.

You can nest one Pblock inside another for hierarchical floorplanning using the **-parent** option as shown in the first example. You can also nest an existing Pblock inside another Pblock using the **set_property** command to define the PARENT property as shown in the second example.

Note: The ISE implementation software does not support extensive use of this feature. Occasionally, map and placement errors occur on designs with nested Pblocks.

Arguments

-parent *arg*- Specifies the name of the parent Pblock to allow creation of nested Pblocks. If the parent is not specified, the default parent of Root is assumed, placing the Pblock at the top-level of the design. You can use the **get_pblocks** command to report currently defined Pblocks that can be used as parents.

Note: If the specified **parent** does not exist an error will be returned

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

name - Specify the name of the Pblock to be created.

Examples

The following example creates a Pblock called Video1 inside another Pblock called Vid_Array:

```
create_pblock -parent Vid_Array Video1
```

The following example creates Pblocks called cpu1 and cpu2, and creates a third Pblock called cpuEngine. Then cpu1 and cpu2 are nested inside cpuEngine using the set_property command:

```
create_pblock cpu1
create_pblock cpu2
create_pblock cpuEngine
set_property PARENT cpuEngine [get_pblocks {cpu1 cpu2}]
```

See Also

- add_cells_to_pblock
- get_pblocks
- place_pblocks
- resize_pblock
- set_property

create_port

Create scalar or bus port

Syntax

```
create_port -direction arg [-from arg] [-to arg] [-diff_pair]
[-interface arg] [-quiet] name
```

Returns

list of port objects that were created

Usage

Name	Optional	Default	Description
-direction	no		Direction of port. Valid arguments are in, out and inout
-from	yes		Beginning index of new bus
-to	yes		Ending index of new bus
-diff_pair	yes		Create differential pair of ports
-interface	yes		Assign new port to this interface
-quiet	yes		Ignore command errors
name	no		Name of the port

Categories

PinPlanning

create_project

Create a new project

Syntax

```
create_project [-part arg] [-force] [-quiet] name [dir]
```

Returns

new project object

Usage

Name	Optional	Default	Description
-part	yes		Target part
-force	yes		Overwrite existing project directory
-quiet	yes		Ignore command errors
name	no		Project name
dir	yes	.	Directory where the project file is saved

Categories

Project

Description

This command creates a PlanAhead project file in the specified directory.

Arguments

name - This argument does not require a parameter name, however, it must appear before the specified *dir*. Since these commands do not have parameters, the PlanAhead tool interprets the first argument as *name* and uses the second argument as *dir*. A project file is created *name*.ppr, and a project data folder is also created *name*.data and both are written into the specified directory *dir*.

The project file created by the PlanAhead tool is an RTL source file by default. You must use the set_property command to set the design_mode property to change the project from an RTL source project to another type of project, such as an I/O Pin Planning project for instance.

dir - This argument specifies the directory name to write the new project file into. If the specified directory does not exist a new directory will be created. If the directory is specified with the complete path, the PlanAhead tool uses the specified path name. However, if *dir* is specified without a path, the PlanAhead tool looks for or creates the directory in the current working directory, or the directory from which the PlanAhead tool was launched.

Note: When creating a project in GUI-mode, the PlanAhead tool appends the filename *name* to the directory name *dir* and creates a project directory with the name *dir*/*name* and places the new project file and project data folder into that project directory.

-part *partName* - specifies the Xilinx part to be used for the project. This can be changed after the project is created. If the **-part** option is not specified, the default part will be used. Currently the default part is xc6vlx75tff484-1.

-force - This option is required to overwrite an existing project. If the project name is already define in the specified *dir* then you must also specify the **-force** option for the PlanAhead tool to overwrite the existing project.

Note: If the existing project is currently open in the PlanAhead tool, the new project will overwrite the existing project on the disk, but both projects will be opened in the PlanAhead tool. In this case you should probably run the **close_project** command prior to running **create_project**.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example creates a project called Project1 in a directory called myDesigns:

```
create_project Project1 myDesigns
```

Note: Because the *dir* is specified as the folder name only, the PlanAhead tool will create the project in the current working directory, or the directory from which the PlanAhead tool was launched.

The following example creates a project called Proj1 in a directory called FPGA in C:/Designs. In addition, the PlanAhead tool will overwrite an existing project if one is found to exist in the specified location. In the second and third lines, the location of **-force** is changed to show the flexibility of argument placement.

```
create_project Proj1 C:/Designs/FPGA -force
-or-
create_project Proj1 -force C:/Designs/FPGA
-or-
create_project -force Proj1 C:/Designs/FPGA
```

Note: In all cases the first argument without a preceding keyword is interpreted as the *name* variable, and the second argument without a preceding keyword is the *dir* variable.

The following example creates a new project called `pin_project`, and then sets the **design_mode** property as required for an I/O Pin Planning project, and finally opens an IO design:

```
create_project pin_project C:/Designs/PinPlanning
set_property design_mode PinPlanning [current_fileset]
open_io_design -name io_1
```

See Also

- current_project
- set_property
- open_io_design

create_property

Create property for class of objects(s)

Syntax

```
create_property [-type arg] [-quiet] name [class]
```

Returns

The property that was created if success, "" if failure

Usage

Name	Optional	Default	Description
-type	yes	string	Type of property to create; valid values are string, int, double, bool
-quiet	yes		Ignore command errors
name	no		Name of property to create
class	yes	string	Object type for which property to create; valid values are design, net, cell, pin, port, pblock

Categories

PropertyAndParameter

Description

Creates a new property of the *type* specified with the user-defined *name* for the specified *class* of objects. The property that is created can be assigned to an object of the specified class with the **set_property** command, but is not automatically associated with all objects of that class.

The **report_property -all** command will not report the newly created property for an object of the specified class until the property has been assigned to that object.

Arguments

-type *arg* - Specifies the type of property to create. There are four allowed property types:

- **string** - Allows the new property to be defined with string values. This is the default value when **-type** is not specified.
- **int** - Allows the new property to be defined with long integer values. If a decimal value is specified for an **int** property type, the PlanAhead tool will return an error.
- **double** - Allows the new property value to be defined with a floating point number.
- **bool** - Allows the new property to be defined as a boolean with a true (1) or false (0) value.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

name - Specifies the name of the the property to be defined. The name is case sensitive.

class - Specifies the class of object to assign the new property to. All objects of the specified class will be assigned the newly defined property. The valid classes are: design, net, cell, pin, port, and pblock.

Examples

The following example defines a property called PURPOSE for cell objects:

```
create_property PURPOSE cell
```

Note: Because the **-type** was not specified, the value will default to strings.

The following example creates a pin property called COUNT which holds an Integer value:

```
create_property -type int COUNT pin
```

See Also

- get_property
- list_property
- list_property_value
- report_property
- reset_property
- set_property

create_reconfig_module

Create and add a new Reconfigurable Module to a cell. The cell will be marked as a Reconfigurable Partition if it is not already.

Syntax

```
create_reconfig_module [-force] [-blackbox] [-quiet] name cell
```

Returns

new reconfigurable module object

Usage

Name	Optional	Default	Description
-force	yes		Run the command, even if there are pending constraint changes, which will be lost
-blackbox	yes		Create a Black Box Reconfigurable Module. Source and constraint files may not be added to a Black Box RM.
-quiet	yes		Ignore command errors
name	no		Name of the new Reconfigurable Module
cell	no		Cell to receive the new Reconfigurable Module

Categories

PartialReconfiguration

create_run

Define a synthesis or implementation run for the current project

Syntax

```
create_run [-constrset arg] [-parent_run arg]
[-part arg] -flow arg [-strategy arg] [-quiet] name
```

Returns

run object

Usage

Name	Optional	Default	Description
-constrset	yes		Constraint fileset to use
-parent_run	yes		Synthesis run to link to new implementation run
-part	yes		Target part
-flow	no		Flow name
-strategy	yes		Strategy to apply to the run
-quiet	yes		Ignore command errors
name	no		Name for new run

Categories

Project

Description

This command defines a synthesis or implementation run for use in the PlanAhead tool. The attributes of the run can be configured with the use of the **set_property** command:

Arguments

-constrset *arg* - Specifies the constraint set to use for the synthesis or implementation run.

-parent_run *arg* - For an RTL sources project, the parent_run must be specified for implementation runs, but is not required for synthesis runs. The parent_run argument identifies which synthesis run the implementation run will implement. For netlist-based projects the parent_run argument is not required to define an implementation run.

-part *partName* - specifies the Xilinx part to be used for the run. If the **-part** option is not specified, the default part defined for the project will be assigned as the part to use.

-flow *arg* - Specify the tool flow and release version for the synthesis tool {XST 13} or the implementation tool {ISE 13}.

-strategy *arg* - Specifies a strategy to employ for the synthesis or implementation run. There are many different strategies to choose from within the PlanAhead tool, including custom strategies you can define. Refer to the *PlanAhead User Guide (ug632)* for a discussion of the available synthesis and implementation strategies. If the strategy argument is not specified, "Synthesis Defaults" or "Implementation Defaults" will be used as appropriate.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

name - Specifies the name of the synthesis or implementation run to be created.

Examples

The following example creates a run named first_pass referencing the XST 13 flow:

```
create_run -flow {XST 13} first_pass
```

Note: The defaults of sources_1, constrs_1, and the default part for the project will be used in the synthesis run. In addition, since this is a synthesis run, the **-parent_run** argument is not required.

The following example creates an implementation run based on the ISE 13 tool flow, and attaches it to the first_pass synthesis run previously created:

```
create_run -flow {ISE 13} -parent_run first_pass imp_1
```

Note: The **-parent_run** argument is required in this example because it is an implementation of synthesized RTL sources.

See Also

- current_run
- launch_runs
- set_property

create_slack_histogram

Create Histogram

Syntax

```
 create_slack_histogram [-to args] [-delay_type arg]
[-num_bins arg] [-slack_less_than arg] [-slack_greater_than arg]
[-group args] [-report_unconstrained] [-significant_digits arg]
[-scale arg] [-name arg] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-to	yes		To clock
-delay_type	yes	max	Type of path delay: Values: max, min, min_max
-num_bins	yes	10	Maximum number of bins: Value =1
-slack_less_than	yes	1e+30	Display paths with slack less than this
-slack_greater_than	yes	-1e+30	Display paths with slack greater than this
-group	yes		Limit report to paths in this group(s)
-report _unconstrained	yes		Report unconstrained end points
-significant_digits	yes	3	Number of digits to display: Range: 0 to 13
-scale	yes	linear	Type of scale on which to draw the histogram; Values: linear, logarithmic
-name	yes		Output the results to GUI panel with this name
-quiet	yes		Ignore command errors

Categories

Report

crossprobe_fed

Crossprobe paths of Bels and Nets to FPGAEditor

Syntax

```
crossprobe_fed [-run arg] [-path args] [-objects args] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-run	yes		Implemented run to launch FED with
-path	yes		Path connecting primitive cells and nets
-objects	yes		List of cells and nets to crossprobe
-quiet	yes		Ignore command errors

Categories

ToolLaunch

Description

This command allows you to cross-probe from the PlanAhead tool to the Xilinx FPGA Editor that was opened with the **launch_fpga_editor** command. You can select both objects and timing paths to cross-probe between the editors.

crossprobe_fed [-run <arg>] [-path <args>] [-objects <args>] [-quiet]

Arguments

-run *arg* - Specify the run name to use when cross-probing.

-path *args* - Specify one or more paths to cross-probe in the FPGA Editor.

-objects *args* - Specify one or more objects to cross-probe in the FPGA Editor. You can use any of the **get_*** commands, such as **get_cells** or **get_ports**, to select objects for cross-probing in the FPGA Editor. See the example below.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example defines a group of objects to be cross-probed in the FPGA Editor using the **get_cells** command to hierarchically locate primitive cells:

```
crossprobe_fed -run impl_1 -objects [get_cells -hier -filter {IS_PRIMITIVE==1}]
```

The following example identifies a path to be cross-probed in the FPGA Editor:

```
crossprobe_fed -path {wbClk i_2090 wbClk_IBUF i_2089 n_0_2089 egressLoop[4].egressFifo/buffer_
```

See Also

- get_cells
- get_ports
- launch_fpga_editor

current_design

Set or get the current design.

Syntax

```
current_design [-quiet] [design]
```

Returns

design object

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
design	yes		Name of current design to be set

Categories

SDC, XDC

Description

This command can either define the current design, or will return the name of the current design in the active project.

The current design and current instance are the target of most Tcl commands, design edits and constraint changes made in the PlanAhead tool. The current instance can be defined using the **current_instance** command.

You can use the **get_designs** command to get a list of open designs in the active project, and use the **get_projects** command to get a list of open projects.

Note: This command returns the current design object.

Arguments

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

design - (Optional) Specifies the name of design to set as the current design. If a *design* is not specified, the command simply returns the current design of the active project.

Example

The following example sets the specified RTL design as the current design:

```
current_design rtl_1
```

See Also

- current_instance
- get_designs
- get_projects

current_fileset

Get the current fileset.

Syntax

```
current_fileset [-constrset] [-quiet]
```

Returns

current fileset (the current srcset by default)

Usage

Name	Optional	Default	Description
-constrset	yes		Get the current constraints fileset
-quiet	yes		Ignore command errors

Categories

Project

Description

The **current_fileset** command is used to obtain the name of the active constraint fileset within a the PlanAhead tool project.

The command returns the name of the currently active source or constraint fileset.

Arguments

-constrset - Returns the name of the currently active constraint set. Without this argument the PlanAhead tool returns the active source fileset by default.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute

Example

The following example returns the name of the active constraint file set.

```
current_fileset -constrset -quiet
```

Note that with the **-quiet** option specified, the PlanAhead tool will not return anything if it encounters an error in processing the command.

current_instance

Set or get the current instance

Syntax

```
current_instance [-quiet] [instance]
```

Returns

instance name

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
instance	yes		Name of instance

Categories

SDC, XDC

Description

This command will set the current instance in the design hierarchy to the specified instance cell or to the top-level cell. The name of the presently defined current instance is returned.

The current design and current instance are the target of most of the commands and design changes made in the PlanAhead tool. The current design can be defined using the **current_design** command.

You must specify the *instance* name relative to the currently defined instance, and use the established hierarchy separator to define instance paths. Use '..' to traverse up the hierarchical instance path.

Note: This command returns the instance object.

Arguments

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

instance - (Optional) Specifies the name of the instance to be set as the current instance of the current design. If the *instance* argument is not specified, the current instance is reset to the top-level cell in the design hierarchy. If the *instance* is specified as '.' then the name of the presently defined current instance is returned, and the instance is not changed.

Note: The hierarchical instance path must be specified using the currently defined hierarchy separator, which can be found with the **get_hierarchy_separator** command

Examples

The following example sets the current instance to the top-level cell of the current design:

```
current_instance
INFO: [PlanAhead-618] Current instance is the top level of design 'rtl_1'.
```

The following example first sets the hierarchy separator character, and then sets the current instance relative to the presently defined current instance:

```
set_hierarchy_separator |
current_instance ..|cpu_iwb_dat_o|buffer_fifo
```

The following example returns the name of the presently defined current instance:

```
current_instance .
cpuEngine|cpu_iwb_dat_o|buffer_fifo
```

See Also

- current_design
- get_hierarchy_separator
- set_hierarchy_separator

current_project

Set or get current project

Syntax

```
current_project [-quiet] [project]
```

Returns

current or newly set project object

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
project	yes		Project to set as current

Categories

Project

Description

This command specifies the current_project or returns the current_project when no project is specified.

Arguments

project - Specifies the name of the PlanAhead project to make current. This command can be used prior to the **close_project** to make a specific project active and then to close the project.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example sets `project_2` as the current project:

```
current_project project_2
```

This command makes the current project the focus of all the PlanAhead tool commands. In the GUI mode, the current_project is defined automatically when switching the GUI between projects.

The following example returns the name of the current project in the PlanAhead tool:

```
current_project
```

Note: The returned value is the name of the PlanAhead project and not the name or path of the project file.

See Also

- close_project
- current_design

current_run

Set or get the current run

Syntax

```
current_run [-synthesis] [-implementation] [-quiet] [run]
```

Returns

run object

Usage

Name	Optional	Default	Description
-synthesis	yes		Set or get the current synthesis run
-implementation	yes		Set or get the current implementation run (default unless '-synthesis' is specified)
-quiet	yes		Ignore command errors
run	yes		Run to set as current; optional

Categories

Project

Description

This command can either define the current synthesis or implementation run, or will return the name of the current run. The current run is the one automatically selected when the Synthesize or Implement commands are launched.

You can use the **get_runs** command to determine the list of defined runs in the current design.

Arguments

-synthesis - Specifies that the **current_run** command should set or return the name of the current synthesis run.

-implementation - Specifies that the **current_run** command should set or return the name of the current implementation run. This is the default used when neither **-synthesis** or **-implementation** are specified.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

run - Specifies the name of the synthesis or implementation run to make the current run.

Examples

The following example defines the first_pass run as the current_run:

```
current_run first_pass
```

Note: The **-synthesis** and **-implementation** arguments are not required because the name allows the PlanAhead tool to identify the specific run of interest.

The following command returns the name of the current implementation run:

```
current_run -implementation -quiet
```

See Also

get_runs

data2mem

Data to memory translation

Syntax

```
data2mem [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors

Categories

ToolLaunch, Simulation

delete_clocknetworks_results

Clear a set of clock networks results from memory

Syntax

```
delete_clocknetworks_results [-quiet] name
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
name	no		Name for the set of results to clear

Categories

Report

delete_debug_core

Delete ChipScope debug core

Syntax

```
delete_debug_core [-quiet] cores ...
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
cores	no		ChipScope debug core

Categories

ChipScope

Description

This command removes ChipScope debug cores from the current project. The debug cores may have been added by the create_debug_core command, or imported by the **read_chipscope_cdc** command. In either case the core will be removed from the current project.

Arguments

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

cores - List of one or more ChipScope debug core names to remove from the current project.

Examples

The following command deletes the myCore debug core from the current project:

```
delete_debug_core myCore
```

The following command deletes all debug cores from the current project:

```
delete_debug_core [get_debug_cores]
```

Note: The **get_debug_cores** command returns all debug cores as a default.

See Also

- create_debug_core
- get_debug_cores
- read_chipscope_cdc

delete_debug_port

Delete ChipScope debug port

Syntax

```
delete_debug_port [-quiet] ports ...
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
ports	no		ChipScope debug port

Categories

ChipScope

Description

Deletes ports from ChipScope debug cores in the current project. You can disconnect a signal from a debug port using **disconnect_debug_port**, or remove the port altogether using this command.

Arguments

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

ports - Specify the core_name/port_name of the debug port to be removed from the core.

Examples

The following example deletes the DATA port from myCore:

```
delete_debug_port myCore/DATA
```

Some ports cannot be deleted because an ILA port requires one CLK port and one TRIG port as a minimum.

The following example deletes the trigger ports (TRIG) from the myCore debug core:

```
delete_debug_port [get_debug_ports myCore/TRIG*]
```

Note: This example will not delete all TRIG ports from myCore, because an ILA core must have at least one TRIG port. The effect of this command will be to delete the TRIG ports starting at TRIG0 and removing all of them except the last port.

See Also

- disconnect_debug_port
- get_debug_ports

delete_fileset

Delete a fileset

Syntax

```
delete_fileset [-quiet] fileset
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
fileset	no		Fileset to be deleted

Categories

Project, Simulation

Description

Deletes the specified fileset. However, if the fileset cannot be deleted, then no message is returned.

Arguments

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

fileset - Specifies the name of the fileset to be deleted. The PlanAhead tool will delete the fileset if it is possible. However, the PlanAhead tool will not delete the last constraint or simulation fileset.

Example

The following example deletes the sim_2 fileset from the current project.

```
delete_fileset sim_2
```

Note: The fileset and all of its files are removed from the project. The files are not removed from the hard drive.

See Also

- create_fileset
- current_fileset

delete_interface

Delete I/O port interfaces from the project

Syntax

```
delete_interface [-all] [-quiet] interfaces ...
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-all	yes		Also delete all of the ports and buses belonging to the interface
-quiet	yes		Ignore command errors
interfaces	no		I/O port interfaces to delete

Categories

PinPlanning

delete_pblock

Remove Pblock

Syntax

```
delete_pblock [-hier] [-quiet] pblocks ...
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-hier	yes		Also delete all the children of Pblock
-quiet	yes		Ignore command errors
pblocks	no		Pblocks to delete

Categories

Floorplan

Description

This command deletes the specified Pblocks from the design. Pblocks are created using the **create_pblock** command.

Arguments

-hier - Specifies that Pblocks nested inside the specified Pblock should also be deleted. If the Parent Pblock is deleted without the **-hier** option specified, the nested Pblocks will simply be moved up one level.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

pblocks - Specify one or more Pblocks to be deleted.

Example

The following example deletes the specified Pblock as well as any Pblocks nested inside:

```
delete_pblock -hier cpuEngine
```

See Also

create_pblock

delete_port

Delete give list of ports or port bus

Syntax

```
delete_port [-quiet] ports ...
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
ports	no		Ports to delete

Categories

PinPlanning

delete_power_results

Set default switching activity on specified types

Syntax

```
delete_power_results -name arg [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-name	no		Name for the set of results to clear
-quiet	yes		Ignore command errors

Categories

XDC

Description

This command clears the power analysis results for the specified results set.

Note: This command operates silently and does not return direct feedback of its operation

Arguments

-name *arg* - (Required) Specifies the name of the results set to clear. This name was either explicitly defined, or was automatically defined when the **report_power** command was run.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example runs power analysis, and then clears the results:

```
report_power -name my_set
delete_power_results -name my_set
```

See Also

- report_power
- reset_switching_activity
- set_switching_activity

delete_reconfig_module

Remove Reconfigurable Module(s)

Syntax

```
delete_reconfig_module [-quiet] reconfig_module
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
reconfig_module	no		Reconfigurable Module(s) to delete

Categories

PartialReconfiguration

delete_rpm

Delete an RPM

Syntax

```
delete_rpm [-quiet] rpm
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
rpm	no		RPM to delete

Categories

Floorplan

Description

This command deletes the specified Relationally Placed Macro (RPM) from the design.

An RPM is a collection of logic elements (FFS, LUT, CY4, RAM, etc.) collected into a set (U_SET, H_SET, and HU_SET). The placement of each element within the set, relative to other elements of the set, is controlled by Relative Location Constraints (RLOCs). Logic elements with RLOC constraints and common set names are associated in an RPM. Refer to the Constraints Guide (UG625) for more information on defining these constraints.

Only user-defined RPMs can be deleted from the design. RPMs defined by the hierarchy or defined in the netlist cannot be deleted by this command.

Arguments

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

rpm - Specify the RPM to be deleted.

Example

The following example deletes the specified RPM from the design:

```
delete_rpm cs_ila_0/U0
```

delete_run

Delete an existing run

Syntax

```
delete_run [-noclean_dir] [-quiet] run
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-noclean_dir	yes		Do not remove all output files and directories from disk
-quiet	yes		Ignore command errors
run	no		Run to modify

Categories

Project

Description

This command will delete a run from the project, and will delete all results of the run from the project directory on the hard drive unless directed otherwise.

Arguments

-noclean_dir - Specifies that the PlanAhead tool should not delete the run results from the hard drive. The run will be deleted from the project, but the run files will remain in the project directory.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

run - Specifies the name of the synthesis or implementation run to delete from the project.

Examples

The following example deletes the first_pass run from the project:

```
delete_run first_pass
```

Note: In this example, all run results will also be removed from the project directory on the hard drive.

The following command deletes the first_pass run, but leaves the run results on the hard drive:

```
delete_run -noclean_dir first_pass
```

See Also

- create_run
- current_run

delete_timing_results

Clear a set of timing results from memory

Syntax

```
delete_timing_results [-type arg] [-quiet] name
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-type	yes		Type of timing results to clear; Values: timing_path, slack_histogram, clock_interaction
-quiet	yes		Ignore command errors
name	no		Name for the set of results to clear

Categories

Report, XDC

demote_run

Unpromote previously promoted Partitions so that they are no longer available for import

Syntax

```
demote_run [-run arg] [-partition_names args]
[-promote_dir arg] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-run	yes		Promoted run to be demoted
-partition_names	yes		List of Partitions to be promoted
-promote_dir	yes		Directory to be demoted
-quiet	yes		Ignore command errors

Categories

PartialReconfiguration, Partition

device_enable

Enable specified devices

Syntax

```
device_enable [-quiet] enable
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
enable	no		Enable the given part

Categories

disconnect_debug_port

Disconnect nets and pins from debug port channels

Syntax

```
disconnect_debug_port [-channel_index arg] [-quiet] port
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-channel_index	yes		Disconnect the net at channel index
-quiet	yes		Ignore command errors
port	no		Debug port name

Categories

ChipScope

Description

Signals from the Netlist Design are connected to ports of a ChipScope debug core by the use of the **connect_debug_port** command. Use this command to disconnect the signals from the debug ports.

A port can also be deleted from the debug core rather than simply disconnected by using the **delete_debug_port** command.

If you need to determine the specific name of a port on a debug core you can use the **get_debug_ports** command to list all ports on a core. You can also use the **report_debug_core** command to list all of the cores in the projects, and their specific parameters.

Arguments

-**channel_index** *value* - Specifies which channel index of a port to disconnect.

Note: The PlanAhead tool will disconnect the entire port if the channel_index is not specified.

port - specifies the name of the port on the debug core to disconnect. The port name must be specified as core_name/port_name. See the examples below.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example disconnects only the specified channel index from the TRIG0 port of myCore:

```
disconnect_debug_port -channel_index 2 myCore/TRIG0
```

If you do not specify the channel_index, all of the channels of the specified port will be disconnected, as in the following example:

```
disconnect_debug_port myCore/TRIG0
```

See Also

- connect_debug_port
- delete_debug_port
- get_debug_ports
- report_debug_core

endgroup

End a set of commands that can be undone/redone as a group

Syntax

```
endgroup [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors

Categories

GUIControl

Description

This command is used to conclude a sequence of commands that can be undone or redone as a series. Use this command after the **startgroup** command to create a sequence of commands.

Arguments

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example defines a startgroup, executes a sequence of related commands, and then executes the endgroup. This sequence of commands can be undone as a group:

```
startgroup
create_pblock pblock_wbArbEngine
create_pblock pblock_usbEng
add_cells_to_pblock pblock_wbArbEngine [get_cells [list wbArbEngine]] -clear_locs
add_cells_to_pblock pblock_usbEng [get_cells [list usbEngine1/usbEngineSRAM]] -clear_locs
endgroup
```

See Also

- redo
- startgroup
- undo

filter

Filter a list, resulting in new list

Syntax

```
filter [-regexp] [-nocase] [-quiet] [objects] [filter]
```

Returns

new list

Usage

Name	Optional	Default	Description
-regexp	yes		Operators =~ and !~ use regular expressions
-nocase	yes		Perform case-insensitive matching (valid only when -regexp specified)
-quiet	yes		Ignore command errors
objects	yes		List of objects to filter
filter	yes		Filter list with expression

Categories

Object, PropertyAndParameter, XDC

Description

This command takes a list of objects, and returns a reduced list of objects that match the specified filter search pattern.

Arguments

-regexp - Indicates that the filter patterns are specified as regular expressions. For more information on defining regular expressions refer to http://wiki.tcl.tk/396, or to http://wiki.tcl.tk/989 for examples.

-nocase - Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

<*objects*> - Specifies a list of objects that should be filtered to reduce the set to the desired results. The list of objects can be obtained by using one of the many **get_*** commands such as get_parts.

<*filter*> - Filter the objects with the specified expression. The specified pattern filters the list of objects returned based on property values on the objects. You can find out what properties are on an object with the report_property or list_property command. Any property/value pair can be used as a filter. In the case of the "part" object, "DEVICE", "FAMILY" and "SPEED" are some of the properties that can be used to filter results.

The specific operators that can be used in the filter expression are ==, !=, and =~, as well as && and || between filter expressions.

Example

The following example returns a list of parts filtered for the specified speed grade:

```
filter [get_parts] {speed == -3}
filter [get_parts] {speed == -3  ||  speed == -2}
```

The second example command filters parts based according to speed grade -3 OR speed grade -2. All parts matching either speed grade will be returned.

The following example uses regular expression and returns a list of VStatus ports in the design, with zero or more wildcards, and the numbers 0 to 9 appearing one or more times within square brackets:

```
filter -regexp [get_ports] {NAME =~ VStatus.*\[[0-9]+\]}
```

See Also

- get_parts
- get_ports

find_top

Find top module candidates in the supplied files, fileset, or active fileset. Returns a rank ordered list of candidates.

Syntax

```
find_top [-fileset arg] [-files args] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-fileset	yes		Fileset to parse to search for top candidates
-files	yes		Files to parse to search for top candidates
-quiet	yes		Ignore command errors

Categories

Project

generate_ip

Generate a configurable IP

Syntax

```
generate_ip [-srcset arg] -files args [-quiet]
```

Returns

list of files that were generated

Usage

Name	Optional	Default	Description
-srcset	yes		Source set name
-files	no		IP source files to be generated
-quiet	yes		Ignore command errors

Categories

COREGenerator

get_cells

Get a list of cells in the current design

Syntax

```
get_cells [-hsc arg] [-hierarchical] [-regexp] [-nocase]
[-filter arg] [-of_objects args] [-match_style arg] [-quiet]
[patterns]
```

Returns

list of cell objects

Usage

Name	Optional	Default	Description
-hsc	yes	/	Hierarchy separator
-hierarchical	yes		Search level-by-level in current instance
-regexp	yes		Patterns are full regular expressions
-nocase	yes		Perform case-insensitive matching (valid only when -regexp specified)
-filter	yes		Filter list with expression
-of_objects	yes		Get cells of these pins or nets
-match_style	yes	sdc	Style of pattern matching, valid values are ucf, sdc
-quiet	yes		Ignore command errors
patterns	yes	*	Match cell names against patterns

Categories

SDC, XDC, Object

Description

This command returns a list of cell objects in the current design that match a specified search pattern. The default command returns a list of all cells in the design.

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error

Arguments

-hsc *arg* - The default hierarchy separator is '/'. Use this argument to specify a different hierarchy separator.

-hierarchical - Get cells from all levels of the design hierarchy. Without this argument, the PlanAhead tool will only return cells from the top-level of the design hierarchy.

-regexp - Specifies that the search patterns are regular expressions.

-nocase - Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by get_cells based on property values on the cells. You can find out what properties are on an object with the report_property or list_property commands. In the case of the "cell" object, "IS_PARTITION", "IS_PRIMITIVE" and "IS_LOC_FIXED" are some of the properties that can be used to filter results.

The specific operators that can be used in the filter expression are ==, !=, and =~, as well as && and || between filter patterns.

-of_objects *arg* - Get the cells connected to the specified pin or net objects.

-match_style [sdc | ucf] - Indicates that the search pattern matches UCF constraints or SDC constraints. The default is SDC.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

patterns - Match cells against the specified patterns. The default pattern is the wildcard '*' which returns all cells in the project. More than one pattern can be specified to find multiple cells based on different search criteria.

Note: You must enclose multiple search patterns in braces {} to present the list as a single element.

Examples

The following example returns a list of properties and property values attached to a specific cell object:

```
report_property [lindex [get_cells] 1]
```

Note: If there are no cells matching the pattern the PlanAhead tool will return a warning indicating that no cells matched the specified pattern.

The following example prints a list of the library cells instantiated into the design at all levels of the hierarchy, sorting the list for unique names so that each cell is only printed one time:

```
foreach cell [lsort -unique [get_property LIB_CELL [get_cells -hier -filter \ {IS_PRIMITIVE==
```

See Also

- get_lib_cells
- get_nets
- get_pins
- list_property
- report_property

get_clocks

Get a list of clocks in the current design

Syntax

```
 get_clocks [-regexp] [-nocase] [-filter arg] [-of_objects args]
[-match_style arg] [-quiet] [patterns]
```

Returns

list of clocks

Usage

Name	Optional	Default	Description
-regexp	yes		Patterns are full regular expressions
-nocase	yes		Perform case-insensitive matching (valid only when -regexp specified)
-filter	yes		Filter list with expression
-of_objects	yes		Get clocks of these pins or nets
-match_style	yes	sdc	Style of pattern matching, valid values are ucf, sdc
-quiet	yes		Ignore command errors
patterns	yes	*	Match clock names against patterns

Categories

SDC, XDC, Object

Description

This command returns a list of clocks in the current design that match a specified search pattern. The default command returns a list of all clocks in the design, like the **all_clocks** command.

Clocks can be created using the **create_clock** or the **create_generated_clock** commands, or can be automatically generated by the tool, at the output of MMCM for instance.

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object to the list is not permitted and will result in a Tcl error

Arguments

-regexp - Specifies that the search patterns are regular expressions.

-nocase - Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of **-regexp** only.

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by **get_clocks** based on property values on the clocks. You can find out what properties are on an object with the **report_property** or **list_property** commands. In the case of the "clock" object, "PERIOD", "WAVEFORM", and "IS_GENERATED" are some of the properties that can be used to filter results.

The specific operators that can be used in the filter expression are ==, !=, and =~, as well as && and || between filter patterns.

-of_objects *args* - Get the clocks connected to the specified pin or net objects.

-match_style [sdc | ucf] - Indicates that the search pattern matches UCF constraints or SDC constraints. The default is SDC.

-include_generated_clocks - Returns all generated clocks that use the specified clock as the source or master clock. This argument should be used when clock *patterns* are specified in order to return generated clocks of the specified master clocks.

Note: You can get just the generated clocks with the **get_generated_clocks** command.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

patterns - Match clocks against the specified patterns. The default pattern is the wildcard '*' which returns all clocks in the project. More than one pattern can be specified to find multiple clocks based on different search criteria.

Note: You must enclose multiple search patterns in braces {} to present the list as a single element.

Examples

The following example returns a list of clocks matching the various search patterns:

```
get_clocks {*clock *ck *Clk}
```

Note: If there are no clocks matching the pattern the PlanAhead tool will return a warning indicating that no cells matched the specified pattern.

The following example returns the master clock object, and all generated clocks derived from specified clock:

```
get_clocks -include_generated_clocks wbClk
```

The following example returns all properties and property values attached to the specified clock:

```
report_property -all [get_clocks wbClk]
```

See Also

- all_clocks
- create_clock
- create_generated_clock
- get_generated_clocks
- list_property
- report_property

get_debug_cores

Get a list of ChipScope debug cores in the current design

Syntax

```
get_debug_cores [-filter arg] [-regexp] [-nocase] [-quiet]
[patterns]
```

Returns

list of debug_core objects

Usage

Name	Optional	Default	Description
-filter	yes		Filter list with expression
-regexp	yes		Patterns are full regular expressions
-nocase	yes		Perform case-insensitive matching (valid only when -regexp specified)
-quiet	yes		Ignore command errors
patterns	yes	*	Match debug cores against patterns

Categories

Object, ChipScope

Description

This command returns a list of Chipscope debug cores in the current project that match a specified search pattern. The default command returns a list of all debug cores in the project.

Debug cores are added to the project with the create_debug_core or the read_chipscope_cdc commands. When a Chipscope debug core is added to the project, it is contained within an ICON controller core, and includes a CLK port and a trigger port (TRIG) as a default. Additional ports can be added to the debug core with the use of the create_debug_port command.

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error

Arguments

-regexp - Specifies that the patterns are regular expressions.

-nocase - Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by **get_debug_cores** based on property values on the parts. You can find out what properties are on an object with the **report_property** or **list_property** commands.

The specific operators that can be used in the filter expression are ==, !=, and =~, as well as && and || between filter expressions.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

patterns - Match parts against the specified patterns. The default pattern is the wildcard '*' which returns all parts. More than one pattern can be specified to find multiple parts based on different search criteria.

Note: You must enclose multiple search patterns in braces {} to present the list as a single element.

Examples

The following command returns a list of the Chipscope debug cores in the current project:

```
get_debug_cores
```

Note: An ICON core is returned as one of the debug cores in the project. You cannot directly create this core, but it is automatically added by the PlanAhead tool when you add any ILA cores to the project.

The following example returns the properties of the specified debug core:

```
report_property [get_debug_cores myCore]
```

The values of the properties returned depend on how the core is configured. You can use the set_property command to configure specific core properties as is shown in the following example:

```
set_property enable_storage_qualification false [get_debug_cores myCore]
```

See Also

- create_debug_core
- create_debug_port
- get_debug_cores
- list_property
- read_chipscope_cdc
- report_property
- set_property

get_debug_ports

Get a list of ChipScope debug ports in the current design

Syntax

```
get_debug_ports [-filter arg] [-regexp] [-nocase] [-quiet]
[patterns]
```

Returns

list of debug_port objects

Usage

Name	Optional	Default	Description
-filter	yes		Filter list with expression
-regexp	yes		Patterns are full regular expressions
-nocase	yes		Perform case-insensitive matching (valid only when -regexp specified)
-quiet	yes		Ignore command errors
patterns	yes	*	Match debug ports against patterns

Categories

Object, ChipScope

Description

This command returns a list of ports defined on Chipscope debug cores in the current project that match a specified search pattern. The default command returns a list of all debug ports in the project.

Debug ports will be defined when Chipscope debug cores are created with the read_chipscope_cdc command, or the create_debug_core command. In addition, ports can be added to existing debug cores with the create_debug_port command.

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error

Arguments

-regexp - Specifies that the patterns are regular expressions.

-nocase - Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters argument filters the list of objects returned by **get_debug_ports** based on property values on the ports. You can find out what properties are on an object with the **report_property** or **list_property** commands. Any property/value pair can be used as a filter. In the case of the "debug_port" object, "PORT_WIDTH", and "MATCH_TYPE" are some of the properties that can be used to filter results.

The specific operators that can be used in the filter expression are ==, !=, and =~, as well as && and || between filter expressions. See the examples below.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

patterns - Match parts against the specified patterns. The default pattern is the wildcard '*' which returns all parts. More than one pattern can be specified to find multiple parts based on different search criteria.

Note: You must enclose multiple search patterns in braces {} to present the list as a single element.

Examples

The following command returns a list of the ports from the Chipscope debug cores in the current project, with a PORT_WIDTH property of 8:

```
get_debug_ports -filter {PORT_WIDTH==8}
```

The following example returns the properties attached to the specified debug port:

```
report_property [get_debug_ports myCore/TRIG0]
```

Note: The debug port is defined by the core_name/port_name combination.

See Also

- create_debug_core
- create_debug_port
- list_property
- read_chipscope_cdc
- report_property

get_default_switching_activity

Get default switching activity on specified default types

Syntax

```
 get_default_switching_activity [-static_probability]
[-toggle_rate] -type args [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-static_probability	yes		report static probability
-toggle_rate	yes		report toggle rate
-type	no		Reports the default seed values of specified types for vector-less propagation engine. List of valid default type values: input, input_set, input_reset, input_enable, register, dsp, bram_read_enable, bram_write_enable, output_enable, clock, all
-quiet	yes		Ignore command errors

Categories

XDC

get_designs

Get a list of designs in the current design

Syntax

```
get_designs [-regexp] [-nocase] [-filter arg] [-quiet]
[patterns]
```

Returns

list of design objects

Usage

Name	Optional	Default	Description
-regexp	yes		Patterns are full regular expressions
-nocase	yes		Perform case-insensitive matching (valid only when -regexp specified)
-filter	yes		Filter list with expression
-quiet	yes		Ignore command errors
patterns	yes	*	Match design names against patterns

Categories

XDC, Object

Description

This command returns a list of open designs in the current project that match a specified search pattern. The default command returns a list of all open designs in the project.

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error

Arguments

-regexp - Specifies that the patterns are regular expressions.

-nocase - Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by get_designs based on property values on the designs. You can find out what properties are on an object with the **report_property** command. In the case of the "design" object, "CONSTRSET", and "PART" are some of the properties that can be used to filter results.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

patterns - Match designs against the specified patterns. The default pattern is the wildcard '*' which returns all designs. More than one pattern can be specified to find multiple designs based on different search criteria.

Examples

The following example returns all open designs in the current project:

```
get_designs
```

The following example returns the assigned properties of an open design matching the search pattern:

```
report_property [get_designs r*]
```

Note: If there are no designs matching the pattern the PlanAhead tool will return a warning indicating that no designs matched the specified pattern.

See Also

report_property

get_files

Get a list of source files

Syntax

```
get_files [-regexp] [-nocase] [-filter arg] [-of_objects args]
[-quiet] [patterns]
```

Returns

list of file objects

Usage

Name	Optional	Default	Description
-regexp	yes		Patterns are full regular expressions
-nocase	yes		Perform case-insensitive matching
-filter	yes		Filter list with expression
-of_objects	yes		Get files of these filesets
-quiet	yes		Ignore command errors
patterns	yes	*	Match file names against patterns

Categories

Object, Project

Description

This command returns a list of files in the current project that match a specified search pattern. The default command returns a list of all files in the project.

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error

Arguments

-regexp - Specifies that the patterns are regular expressions

-nocase - Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by get_files based on property values on the files. You can find out what properties are on an object with the report_property command. Any property/value pair can be used as a filter. In the case of the "file" object, "FILE_TYPE", and "IS_ENABLED" are some of the properties that can be used to filter results.

-of_objects *args* - Specifies one or more filesets to search for the files. The default is to search all filesets.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

patterns - Match files against the specified patterns. The default pattern is the wildcard '*' which returns all files in the project or of_objects. More than one pattern can be specified to find multiple files based on different search criteria.

Examples

The following example returns a list of all files in the current project:

```
get_files
```

The following example returns a list of the Verilog files (*.v) found in the constrs_1 and sim_1 filesets:

```
get_files -of_objects {constrs_1 sim_1} *.v
```

Note: If there are no files matching the pattern the PlanAhead tool will return a warning indicating that no files matched the specified pattern.

See Also

report_property

get_filesets

Get a list of filesets in the current project

Syntax

```
get_filesets [-regexp] [-nocase] [-filter arg] [-quiet]
[patterns]
```

Returns

list of fileset objects

Usage

Name	Optional	Default	Description
-regexp	yes		Patterns are full regular expressions
-nocase	yes		Perform case-insensitive matching
-filter	yes		Filter list with expression
-quiet	yes		Ignore command errors
patterns	yes	*	Match fileset names against patterns

Categories

Object, Project

Description

This command returns a list of filesets in the current project that match a specified search pattern. The default command returns a list of all filesets in the project.

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error

Arguments

-regexp - Patterns are full regular expressions

-nocase - Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by get_filesets based on property values on the filesets. You can find out what properties are on an object with the report_property command.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

patterns - Match fileset names against the specified patterns. The default pattern is the wildcard '*' which returns all filesets. More than one pattern can be specified to find filesets based on multiple search criteria.

Examples

The following example returns all filesets in the current project:

```
get_filesets
```

The following example makes `project_2` the active project, and then gets a list of filesets beginning with the letter s or the letter r:

```
current_project project_2
get_filesets s* r* -quiet
```

Note: If there are no filesets matching the pattern, such as r*, the PlanAhead tool will return a warning indicating that no filesets matched the specified pattern. However, in the above example the use of **-quiet** will suppress that warning message.

The following example looks for filesets beginning with the letter C, using a case insensitive regular expression:

```
get_filesets C.* -regexp -nocase
```

In the above example, constrs_1 and constrs_2 constraint sets would be returned if defined in the current project.

See Also

report_property

get_generated_clocks

Get a list of generated clocks in the current design

Syntax

```
get_generated_clocks [-regexp] [-nocase] [-filter arg] [-quiet]
[patterns]
```

Returns

list of clocks

Usage

Name	Optional	Default	Description
-regexp	yes		Patterns are full regular expressions
-nocase	yes		Perform case-insensitive matching (valid only when -regexp specified)
-filter	yes		Filter list with expression
-quiet	yes		Ignore command errors
patterns	yes	*	Match generated clock names against patterns

Categories

XDC, Object

Description

This command returns a list of generated clocks in the current project that match a specified search pattern. The default command returns a list of all generated clocks in the project.

A generated clock can be added to the design using the create_generated_clock command, or can be automatically generated by the tool, at the output of MMCM for instance.

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error

Arguments

-regexp - Specifies that the search patterns are regular expressions.

-nocase - Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by **get_generated_clocks** based on property values on the clocks. You can find out what properties are on an object with the **report_property** command. In the case of the "generated_clock" object, "DUTY_CYCLE" and "MASTER_CLOCK" are some of the properties that can be used to filter results.

The specific operators that can be used in the filter expression are ==, !=, and =~, as well as && and || between filter patterns.

-of_objects *args* - Specifies one or more pins or nets to that the generated clocks are assigned to.

-match_style [sdc | ucf] - Indicates that the search pattern matches UCF constraints or SDC constraints. The default is SDC.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

patterns - Match generated clocks against the specified patterns. The default pattern is the wildcard '*' which returns all generated clocks in the project.

Example

The following example returns the names of all generated clocks in the current project:

```
get_generated_clocks
```

See Also

- create_generated_clock
- get_clocks
- report_property

get_hierarchy_separator

Get hierarchical separator character

Syntax

```
get_hierarchy_separator [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors

Categories

SDC, XDC

Description

This command returns the character that is currently used for separating levels of hierarchy in the design. You can define the hierarchy separator using the **set_hierarchy_separator** command.

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error

Arguments

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example returns the currently defined hierarchy separator:

```
get_hierarchy_separator
```

See Also

set_hierarchy_separator

get_interfaces

Get a list of I/O port interfaces in the current design

Syntax

```
get_interfaces [-regexp] [-nocase] [-quiet] [patterns]
```

Returns

list of interface objects

Usage

Name	Optional	Default	Description
-regexp	yes		pattern is a regular expression
-nocase	yes		pattern matching should be case insensitive
-quiet	yes		Ignore command errors
patterns	yes		pattern to match for finding I/O port interfaces

Categories

XDC, Object

Description

This command returns a list of IO interfaces in the current project that match a specified search pattern. The default command returns a list of all IO interfaces in the project.

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error

Arguments

-regexp - Specifies that the search patterns are regular expressions.

-nocase - Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by get_generated_clocks based on property values on the clocks. You can find out what properties are on an object with the report_property command.

-of_objects *args* - Specifies one or more pins or nets to that interfaces are assigned to.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

patterns - Match interfaces against the specified patterns. The default pattern is the wildcard '*' which returns all interfaces in the project.

Example

The following example returns a list of all interfaces in the project:

```
get_interfaces
```

See Also

- create_interface
- delete_interface

get_iobanks

Get a list of iobanks.

Syntax

```
 get_iobanks [-regexp] [-nocase] [-filter arg]
[-of_objects args] [-quiet] [patterns]
```

Returns

iobanks

Usage

Name	Optional	Default	Description
-regexp	yes		Patterns are regular expressions.
-nocase	yes		Patterns are case insensitive.
-filter	yes		Filter objects based on property expression.
-of_objects	yes		Get the iobanks of these package_pins.
-quiet	yes		Ignore command errors
patterns	yes	*	Return the iobanks whose names match any one of these patterns.

Categories

XDC, Object

Description

This command returns a list of I/O Banks on the target device in the current project that match a specified search pattern. The default command returns a list of all I/O Banks on the target device.

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error

Arguments

-regexp - Specifies that the search patterns are regular expressions.

-nocase - Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by get_iobanks based on property values on the I/O Banks. You can find out what properties are on an object with the report_property command. In the case of the "iobank" object, "DCI_CASCADE", and "INTERNAL_VREF" are some of the properties that can be used to filter results.

The specific operators that can be used in the filter expression are ==, !=, and =~, as well as && and || between filter patterns.

-of_objects *arg* - Get the iobanks connected to the specified package pins.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

patterns - Match I/O Banks against the specified patterns. The default pattern is the wildcard '*' which returns all I/O Banks in the project.

Example

The following example returns all I/O Banks in the target device:

```
get_iobanks
```

See Also

- get_package_pins
- report_property

get_lib_cells

Get a list of library cells. By default, returns all lib cells associated with the current_design's part

Syntax

```
 get_lib_cells [-regexp] [-nocase] [-filter arg]
[-of_objects args] [-quiet] [patterns]
```

Returns

list of library cells

Usage

Name	Optional	Default	Description
-regexp	yes		Patterns are full regular expressions
-nocase	yes		Perform case-insensitive matching (valid only when -regexp specified)
-filter	yes		Filter list with expression
-of_objects	yes		Get library cells of cells/insts, or libpins
-quiet	yes		Ignore command errors
patterns	yes	*	Match lib cell names against patterns; Pattern should explicitly specify library name e.g. libname /*, you can find library names using 'get_libs' command

Categories

SDC, XDC, Object

Description

Get a list of cells in the library for the target part of the current design. Use this command to query and look for a specific library cell, or type of cell and to get the properties of the cells.

This command requires a hierarchical name which includes the library name as well as the cell name: lib_name/cell_name.

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error

Arguments

-regexp - Specifies that the search patterns are regular expressions.

-nocase - Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by get_cells based on property values on the cells. You can find out what properties are on an object with the report_property or the list_property command.

The specific operators that can be used in the filter expression are ==, !=, and =~, as well as && and || between filter expressions.

-of_objects *arg* - Get library cells of specific instances (cells/insts), or library pins (get_lib_pins).

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

patterns - Match library cells against the specified patterns. The pattern must specify both the library name and the cell name.

Examples

The following example returns the number of the cells in the library for the target part in the current design, and then returns the number of AND type cells in that library:

```
llength [get_lib_cells [get_libs]/*]
795
llength [get_lib_cells [get_libs]/AND*]
18
```

The following example returns the library cell for the specified cell object:

```
get_lib_cells -of_objects [lindex [get_cells] 1]
```

See Also

- get_cells
- get_libs
- get_lib_pins
- list_property
- report_property

get_lib_pins

Create list of library cell pins

Syntax

```
 get_lib_pins [-regexp] [-nocase] [-filter arg]
[-of_objects args] [-quiet] [patterns]
```

Returns

list of library cell pins

Usage

Name	Optional	Default	Description
-regexp	yes		Patterns are full regular expressions
-nocase	yes		Perform case-insensitive matching (valid only when -regexp specified)
-filter	yes		Filter list with expression
-of_objects	yes		Get library cell pins these pins or libcells
-quiet	yes		Ignore command errors
patterns	yes	*	Match lib cell names against patterns

Categories

SDC, XDC, Object

Description

This command returns a list of the pins on a specified cell of the cell library for the target part in the current design.

Note: This command requires a hierarchical name which includes the library name and cell name as well as the pins: lib_name/cell_name/pins

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error

Arguments

-regexp - Specifies that the search patterns are regular expressions.

-nocase - Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by get_lib_pins based on property values on the pins. You can find out what properties are on an object with the report_property or the list_property command.

The specific operators that can be used in the filter expression are ==, !=, and =~, as well as && and || between filter expressions.

-of_objects *arg* - Get library cell pins of the specified pin objects or or library cells (get_lib_cells).

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

patterns - Match lib pins against the specified patterns. The pattern must specify the library name, cell name, and the pins.

Examples

The following example returns all library cell pins:

```
get_lib_pins xt_virtex6/AND2/*
```

The following example returns all pins, of all cells in the cell library for the target device:

```
get_lib_pins [get_libs]/*/*
```

See Also

- get_libs
- get_lib_cells
- list_property
- report_property

get_libs

Create list of libraries

Syntax

```
get_libs [-regexp] [-nocase] [-filter arg] [-of_objects args]
[-quiet] [patterns]
```

Returns

list of libraries

Usage

Name	Optional	Default	Description
-regexp	yes		Patterns are full regular expressions
-nocase	yes		Perform case-insensitive matching (valid only when -regexp specified)
-filter	yes		Filter list with expression
-of_objects	yes		Get library of these libcells
-quiet	yes		Ignore command errors
patterns	yes	*	Match library names against patterns

Categories

SDC, XDC, Object

Description

This command returns the cell library for the target device in the current design. There is a library for each device family because there are primitives that may be available in one device family but not in others.

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error

Arguments

-regexp - Specifies that the search patterns are regular expressions.

-nocase - Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by get_libs based on property values on the libs. You can find out what properties are on an object with the report_property or the list_property command.

-of_objects *arg* - Get libraries of the specified object.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

patterns - Match libraries against the specified patterns. The default pattern is the wildcard '*' which returns all libraries in the project.

Example

The following example returns the cell library for the target part:

```
get_libs
```

See Also

- get_lib_cells
- get_lib_pins
- list_property
- report_property

get_msg_count

Get message count

Syntax

```
get_msg_count [-severity arg] [-id arg] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-severity	yes	ALL	Message severity to query (not valid with -id,) e.g. "ERROR" or "CRITICAL WARNING"
-id	yes		Unique message id to be queried (not valid with -severity,) e.g Common-99
-quiet	yes		Ignore command errors

Categories

Report

Description

This command returns the number of messages that the PlanAhead tool has posted since being invoked. This can give you an idea of how close to the message limit the PlanAhead tool may be getting. You can check the current message limit with the **get_msg_limit** command. You can change the message limit with the **set_msg_limit** command.

By default this command returns the message count for all messages. You can also get the count of a specific severity of message, or for a specific message ID.

Arguments

-id *value* - is found in the PlanAhead tool in the Messages view or other reports when the message is reported. Use the specific message ID for the message of interest for use in this command.

-severity *value* - Specifies the severity of the message. There are five message severities:

- **ERROR** - An ERROR condition implies an issue has been encountered which will render design results unusable and cannot be resolved without user intervention.
- **{CRITICAL WARNING}** - A CRITICAL WARNING message indicates that certain input/constraints will either not be applied or are outside the best practices for a FPGA family. User action is strongly recommended. Note: Since this is a two word value, it must be enclosed in {}.
- **WARNING** - A WARNING message indicates that design results may be sub-optimal because constraints or specifications may not be applied as intended. User action may be taken or may be reserved.
- **INFO** - An INFO message is the same as a STATUS message, but includes a severity and message ID tag. An INFO message includes a message ID to allow further investigation through answer records if needed.
- **STATUS** - A STATUS message communicates general status of the process and feedback to the user regarding design processing. A STATUS message does not include a message ID.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example returns the message count for all messages:

```
get_msg_count -severity ALL
get_msg_count
```

Note: Both lines return the same thing since the default is to return the count for all messages when -severity or -id is not specified.

The following example returns the message count of the specified message ID:

```
get_msg_count -id Netlist-1129
```

See Also

- get_msg_limit
- set_msg_limit

get_msg_limit

Get message limit, return message limit

Syntax

```
get_msg_limit [-severity arg] [-id arg] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-severity	yes	ALL	Message severity to query (not valid with -id,) e.g. "ERROR" or "CRITICAL WARNING"
-id	yes		Unique message id to be queried (not valid with -severity,) e.g Common-99
-quiet	yes		Ignore command errors

Categories

Report

Description

This command returns the number of messages that will be reported by the PlanAhead application while invoked. When the PlanAhead application reaches the defined message limit, it stops reporting messages. The default value is 4,294,967,295. This default value can be changed with the **set_msg_limit** command.

By default this command returns the message limit for all messages. You can also get the limit of a specific severity of message, or for a specific message ID. For instance the following are two messages returned by the PlanAhead application under different circumstances:

INFO: [common-99] This is an example INFO message

CRITICAL WARNING: [Netlist-1129] This message is a CRITICAL WARNING and requires user attention

Note: You can change the severity of a specific message ID with the **set_msg_severity** command.

Arguments

-id *value* - is found in the PlanAhead application in the Messages view or other reports when the message is reported. Use the specific message ID for the message of interest for use in this command. The message IDs above are common-99 and Netlist-1129.

-severity *value* - Specifies the severity of the message. There are five message severities:

- **ERROR** - An ERROR condition implies an issue has been encountered which will render design results unusable and cannot be resolved without user intervention.
- **{CRITICAL WARNING}** - A CRITICAL WARNING message indicates that certain input/constraints will either not be applied or are outside the best practices for a FPGA family. User action is strongly recommended. Note: Since this is a two word value, it must be enclosed in {}.
- **WARNING** - A WARNING message indicates that design results may be sub-optimal because constraints or specifications may not be applied as intended. User action may be taken or may be reserved.
- **INFO** - An INFO message is the same as a STATUS message, but includes a severity and message ID tag. An INFO message includes a message ID to allow further investigation through answer records if needed.
- **STATUS** - A STATUS message communicates general status of the process and feedback to the user regarding design processing. A STATUS message does not include a message ID.

-quiet - This option tells the PlanAhead application to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example returns the limit for CRITICAL WARNING messages:

```
get_msg_limit -severity {CRITICAL WARNING}
```

The default when -severity or -id is not specified is to return the limit for all messages.

The following example returns the message limit of the specified message ID:

```
get_msg_limit -id Netlist-1129
```

See Also

- set_msg_limit
- set_msg_severity

get_nets

Get a list of nets in the current design

Syntax

```
get_nets [-hsc arg] [-hierarchical] [-regexp] [-nocase]
[-filter arg] [-of_objects args] [-match_style arg] [-quiet]
[patterns]
```

Returns

list of net objects

Usage

Name	Optional	Default	Description
-hsc	yes	/	Hierarchy separator
-hierarchical	yes		Search level-by-level in current instance
-regexp	yes		Patterns are full regular expressions
-nocase	yes		Perform case-insensitive matching (valid only when -regexp specified)
-filter	yes		Filter list with expression
-of_objects	yes		Get nets of these pins/ports,cells or clocks
-match_style	yes	sdc	Style of pattern matching, valid values are ucf, sdc
-quiet	yes		Ignore command errors
patterns	yes	*	Match net names against patterns

Categories

SDC, XDC, Object

Description

This command returns a list of nets in the current design that match a specified search pattern. The default command returns a list of all nets in the design.

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error

Arguments

-hsc *arg* - The default hierarchy separator is '/'. Use this argument to specify a different hierarchy separator.

-hierarchical - Get nets from all levels of the design hierarchy. Without this argument, PlanAhead will only return nets from the top-level of the design hierarchy.

-regexp - Specifies that the search patterns are regular expressions.

-nocase - Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by get_nets based on property values on the nets. You can find out what properties are on an object with the report_property or list_property commands. In the case of the "nets" object, "PARENT", "TYPE" and "MARK_DEBUG" are some of the properties that can be used to filter results.

The specific operators that can be used in the filter expression are ==, !=, and =~, as well as && and || between filter patterns.

-of_objects *arg* - Get the nets connected to the specified cell, pin, port, or clock.

-match_style [sdc | ucf] - Indicates that the search pattern matches UCF constraints or SDC constraints. The default is SDC.

-top_net_of_hierarchical_group - Returns the top-level net segments of a hierarchical net. Use this argument to return the top-level net name from a lower-level net segment, or to return the top-level net segments of all nets.

-segments - Return all the segments of a hierarchical net, across all levels of the hierarchy. Note that this differs from the **-hierarchical** argument in that it returns all segments of the specified net, rather than just the specified net.

-boundary_type - Valid values are upper, lower, or both. The default value is upper. This argument returns the net segment at the level (upper) of a specified hierarchical pin, at the level below (lower) the pin or port, or both the level of and the level below.

Note: This argument must be used with the **-of_objects** argument to specify the hierarchical pin.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

patterns - Match nets against the specified patterns. The default pattern is the wildcard '*' which returns all nets in the project. More than one pattern can be specified to find multiple nets based on different search criteria.

Note: You must enclose multiple search patterns in braces {} to present the list as a single element.

Examples

The following example returns a list of nets attached to the specified cell:

```
get_nets -of_objects [lindex [get_cells] 1]
```

Note: If there are no nets matching the pattern the PlanAhead tool will return a warning indicating that no nets matched the specified pattern.

The following example returns a list of nets that have been marked for debug with the connect_debug_port command:

```
get_nets -hier -filter {MARK_DEBUG==1}
```

See Also

- connect_debug_port
- get_cells
- get_clocks
- get_pins
- get_ports
- list_property
- report_property

get_operating_conditions

Get operating conditions values for power estimation

Syntax

```
get_operating_conditions [-voltage args] [-grade] [-process]
[-junction_temp] [-ambient_temp] [-thetaja] [-thetasa] [-airflow]
[-heatsink] [-thetajb] [-board] [-board_temp] [-board_layers]
[-all] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-voltage	yes		Gets voltage value. Supported voltage supplies vary by family.
-grade	yes		Gets temperature grade
-process	yes		Gets process
-junction_temp	yes		Gets Junction Temperature
-ambient_temp	yes		Gets Ambient Temperature
-thetaja	yes		Gets ThetaJA
-thetasa	yes		Gets ThetaSA
-airflow	yes		Gets Airflow
-heatsink	yes		Gets dimensions of heatsink
-thetajb	yes		Gets ThetaJB
-board	yes		Gets Board type
-board_temp	yes		Gets Board Temperature
-board_layers	yes		Gets Board layers
-all	yes		Gets all operating conditions listed in this command line
-quiet	yes		Ignore command errors

Categories

SDC, XDC

get_package_pins

Get a list of package pins

Syntax

```
 get_package_pins [-regexp] [-nocase] [-filter arg]
[-of_objects args] [-quiet] [patterns]
```

Returns

list of package pin objects

Usage

Name	Optional	Default	Description
-regexp	yes		Patterns are regular expressions.
-nocase	yes		Patterns are case insensitive.
-filter	yes		Filter objects based on property expression.
-of_objects	yes		Get the list of package pin objects of these sites iobanks.
-quiet	yes		Ignore command errors
patterns	yes	*	Return the list of package pin objects whose names match any one of these patterns.

Categories

XDC, Object

Description

This command returns a list of the pins on the selected package for the target device. The default command returns a list of all pins on the package.

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error

Arguments

-regexp - Specifies that the search patterns are regular expressions.

-nocase - Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by get_package_pins based on property values on the cells. You can find out what properties are on an object with the report_property command. In the case of the "cell" object, "IS_CLK_CAPABLE", "IS_VREF" and "IS_GLOBAL_CLK" are some of the properties that can be used to filter results.

The specific operators that can be used in the filter expression are ==, !=, and =~, as well as && and || between filter patterns.

-of_objects *arg* - Get the package pins connected to the specified site or iobank objects.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

patterns - Match pins against the specified patterns. The default pattern is the wildcard '*' which returns all pins on the package. More than one pattern can be specified to find multiple pins based on different search criteria.

Examples

The following example returns a list of all pins on the package of the target device:

```
get_package_pins
```

The following example returns the number of clock capable (CC) pins on the package:

```
llength [get_package_pins -filter {IS_CLK_CAPABLE==1}]
```

Note: If there are no pins matching the pattern the PlanAhead tool will return a warning indicating that no pins matched the specified pattern.

See Also

- get_iobanks
- get_sites
- list_property
- report_property

get_param

Get a parameter value

Syntax

```
get_param [-quiet] name
```

Returns

parameter value

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
name	no		Parameter name

Categories

PropertyAndParameter

Description

This command returns the currently defined value for a specified the PlanAhead tool parameter. These parameters are user-definable configuration settings that control various behaviors within the tool. Refer to **report_param** for a description of what each parameter configures or controls.

Arguments

name - Specifies the name of the parameter to get the value of. The list of user-definable parameters can be obtained with **list_param**. This command requires the full name of the desired parameter. It does not perform any pattern matching, and accepts only one parameter at a time.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example returns the current value of the specified parameter:

```
get_param tcl.statsThreshold
```

See Also

- list_param
- report_param
- reset_param
- set_param

get_parts

Get a list of parts available in the software

Syntax

```
get_parts [-regexp] [-nocase] [-filter arg] [-quiet] [patterns]
```

Returns

list of part objects

Usage

Name	Optional	Default	Description
-regexp	yes		Patterns are full regular expressions
-nocase	yes		Perform case-insensitive matching
-filter	yes		Filter list with expression
-quiet	yes		Ignore command errors
patterns	yes	*	Match part names against patterns

Categories

Object, Project

Description

This command returns a list of parts in the current project that match a specified search pattern. The default command returns a list of all parts in the project.

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error

Arguments

-regexp - Specifies that the patterns are specified as regular expressions.

-nocase - Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by get_parts based on property values on the parts. You can find out what properties are on an object with the report_property or list_property command. Any property/value pair can be used as a filter. In the case of the "part" object, "DEVICE", "FAMILY" and "SPEED" are some of the properties that can be used to filter results.

The specific operators that can be used in the filter expression are ==, !=, and =~, as well as && and || between filter expressions. See the examples below.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

patterns - Match parts against the specified patterns. The default pattern is the wildcard '*' which returns all parts. More than one search pattern can be specified to find parts based on different search criteria.

Note: You must enclose multiple search patterns in braces {} to present the list as a single element.

Examples

The following example returns the list of 7vx485t parts, speed grade -1:

```
get_parts -filter {DEVICE == 7vx485t && speed == -1}
```

The following example returns the number of 7 series and 6 series Virtex parts:

```
llength [get_parts -regexp {xc7v.* xc6V.*} -nocase]
```

Note: If there are no parts matching the pattern the PlanAhead tool will return a warning indicating that no parts matched the specified pattern.

See Also

- list_property
- report_property

get_path_groups

Get a list of path groups in the current design

Syntax

```
get_path_groups [-regexp] [-nocase] [-quiet] [patterns]
```

Returns

list of path groups

Usage

Name	Optional	Default	Description
-regexp	yes		Patterns are full regular expressions
-nocase	yes		Perform case-insensitive matching (valid only when -regexp specified)
-quiet	yes		Ignore command errors
patterns	yes	*	Match path group names against patterns

Categories

XDC, Object

Description

This command returns a list of timing path groups in the current project that match a specified search pattern. The default command returns a list of all path groups in the design.

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error

Arguments

-regexp - Specifies that the search patterns are regular expressions.

-nocase - Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

patterns - Match path groups against the specified patterns. The default pattern is the wildcard '*' which returns all path groups in the project.

Examples

The following example returns a list of all the timing path groups in the design.

```
get_path_groups
```

The following example returns all timing path groups with the string "Clk" somewhere in the name:

```
get_path_groups *Clk*
```

Note: If no path groups match the pattern the PlanAhead tool will return a warning.

get_pblocks

Get a list of Pblocks in the current design

Syntax

```
get_pblocks [-regexp] [-nocase] [-quiet] [patterns]
```

Returns

list of pblock objects

Usage

Name	Optional	Default	Description
-regexp	yes		pattern is a regular expression
-nocase	yes		pattern matching should be case insensitive
-quiet	yes		Ignore command errors
patterns	yes		pattern to match for finding Pblocks

Categories

Object, Floorplan

Description

This command returns a list of Pblocks defined in the current project that match a specific pattern. The default command returns a list of all Pblocks in the project.

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error

Arguments

-regexp - Specifies that the search patterns are regular expressions.

-nocase - Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by get_cells based on property values on the cells. You can find out what properties are on an object with the report_property or list_property commands.

-of_objects *arg* - Get the Pblocks that specified cells are assigned to.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

patterns - Match Pblocks against the specified patterns. The default pattern is the wildcard '*' which returns all Pblocks in the project.

Example

The following example returns a list of all Pblocks in the current project:

```
get_pblocks
```

See Also

- create_pblock
- get_cells

get_pins

Get a list of pins in the current design

Syntax

```
get_pins [-hsc arg] [-hierarchical] [-regexp] [-nocase] [-leaf]
[-filter arg] [-of_objects args] [-match_style arg] [-quiet]
[patterns]
```

Returns

list of pin objects

Usage

Name	Optional	Default	Description
-hsc	yes	/	Hierarchy separator
-hierarchical	yes		Search level-by-level in current instance
-regexp	yes		Patterns are full regular expressions
-nocase	yes		Perform case-insensitive matching (valid only when -regexp specified)
-leaf	yes		Get leaf/global pins of nets with -of_objects
-filter	yes		Filter list with expression
-of_objects	yes		Get pins of these cells, nets or clocks
-match_style	yes	sdc	Style of pattern matching, valid values are ucf, sdc
-quiet	yes		Ignore command errors
patterns	yes	*	Match pin names against patterns

Categories

SDC, XDC, Object

Description

This command returns a list of pin objects in the current design that match a specified search pattern. The default command returns a list of all pins in the design.

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error

Arguments

-hsc *arg* - The default hierarchy separator is '/'. Use this argument to specify a different hierarchy separator.

-hierarchical - Get pins from all levels of the design hierarchy. Without this argument, the PlanAhead tool will only return pins from the top-level of the design hierarchy.

-regexp - Specifies that the search patterns are regular expressions.

-nocase - Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

-leaf - Include leaf pins for the parameters specified with the -of_object argument

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by get_pins based on property values on the pins. You can find out what properties are on an object with the report_property or list_property commands. In the case of the "pins" object, "PARENT" and "TYPE" are some of the properties that can be used to filter results.

The specific operators that can be used in the filter expression are ==, !=, and =~, as well as && and || between filter patterns.

-of_objects *arg* - Get the pins connected to the specified cell, pin, port, or clock.

-match_style [sdc | ucf] - Indicates that the search pattern matches UCF constraints or SDC constraints. The default is SDC.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

patterns - Match pins against the specified patterns. The default pattern is the wildcard '*' which returns all pins in the project. More than one pattern can be specified to find multiple pins based on different search criteria.

Note: You must enclose multiple search patterns in braces {} to present the list as a single element.

Example

The following example returns a list of pins attached to the specified cell:

```
get_pins -of_objects [lindex [get_cells] 1]
```

Note: If there are no pins matching the pattern the PlanAhead tool will return a warning indicating that no pins matched the specified pattern.

See Also

- list_property
- report_property

get_ports

Get a list of ports in the current design

Syntax

```
 get_ports [-regexp] [-nocase] [-filter arg] [-of_objects args]
[-match_style arg] [-quiet] [patterns]
```

Returns

list of port objects

Usage

Name	Optional	Default	Description
-regexp	yes		Patterns are full regular expressions
-nocase	yes		Perform case-insensitive matching (valid only when -regexp specified)
-filter	yes		Filter list with expression
-of_objects	yes		Get ports of these nets, clocks
-match_style	yes	sdc	Style of pattern matching, valid values are ucf, sdc
-quiet	yes		Ignore command errors
patterns	yes	*	Match port names against patterns

Categories

SDC, XDC, Object

Description

This command returns a list of port objects in the current design that match a specified search pattern. The default command returns a list of all ports in the design.

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error

Arguments

-regexp - Specifies that the search patterns are regular expressions.

-nocase - Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by get_ports based on property values on the ports. You can find out what properties are on an object with the report_property or list_property commands. In the case of the "ports" object, "PARENT" and "TYPE" are some of the properties that can be used to filter results.

The specific operators that can be used in the filter expression are ==, !=, and =~, as well as && and || between filter patterns.

-of_objects *arg* - Get the ports connected to the specified cell, pin, port, or clock.

-match_style [sdc | ucf] - Indicates that the search pattern matches UCF constraints or SDC constraints. The default is SDC.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

patterns - Match ports against the specified patterns. The default pattern is the wildcard '*' which returns all ports in the project. More than one pattern can be specified to find multiple ports based on different search criteria.

Note: You must enclose multiple search patterns in braces {} to present the list as a single element.

Example

The following example returns a list of pins attached to the specified cell:

```
get_ports -of_objects [lindex [get_cells] 1]
```

Note: If there are no ports matching the pattern the PlanAhead tool will return a warning indicating that no ports matched the specified pattern.

See Also

- list_property
- report_property

get_projects

Get a list of projects

Syntax

```
get_projects [-regexp] [-nocase] [-filter arg] [-quiet]
[patterns]
```

Returns

list of project objects

Usage

Name	Optional	Default	Description
-regexp	yes		Patterns are full regular expressions
-nocase	yes		Perform case-insensitive matching (valid only when -regexp specified)
-filter	yes		Filter list with expression
-quiet	yes		Ignore command errors
patterns	yes	*	Match project names against patterns

Categories

Object, Project

Description

This command returns a list of open projects that match the specified search pattern. The default returns a list of all open projects.

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error

Arguments

-regexp - Specifies that the patterns are regular expressions.

-nocase - Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by get_nets based on property values on the nets. You can find out what properties are on an object with the report_property or list_property commands. In the case of the "projects" object, "NAME", "DIRECTORY" and "TARGET_LANGUAGE" are some of the properties that can be used to filter results.

The specific operators that can be used in the filter expression are ==, !=, and =~, as well as && and || between filter patterns.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

patterns - Match projects against the specified patterns. The default pattern is the wildcard '*' which returns all parts. More than one pattern can be specified to find multiple projects based on different search criteria.

Examples

The following example returns a list of all open projects.

```
get_projects
```

The following example sets a variable called project_found to the length of the list of projects returned by get_projects, then prints either that projects were found or were not found as appropriate:

```
set project_found [llength [get_projects ISC*] ]
if {$project_found > 0} {puts "Project Found."} else {puts "No Projects Found."}
```

Note: If there are no projects matching the pattern PlanAhead will return a warning indicating that no projects matched the specified pattern.

See Also

- create_project
- current_project
- open_project

get_property

Get properties of object

Syntax

```
get_property [-quiet] property_name object
```

Returns

property value

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
property_name	no		Name of property whose value is to be retrieved
object	no		Object to query for properties

Categories

XDC, Object, PropertyAndParameter

Description

Returns the current value of the named property from the specified object. If the property is not currently assigned to the object, or is assigned without a value, then the **get_property** command returns nothing.

Arguments

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

name - Specifies the name of the the property to be returned. The name is not case sensitive.

object - Specifies an object to query.

Example

The following example returns the NAME property from the specified cell:

```
get_property NAME [lindex [get_cells] 3]
```

See Also

- create_property
- get_cells
- list_property
- list_property_value
- report_property
- reset_property
- set_property

get_reconfig_modules

Get a list of Reconfigurable Modules in the current project

Syntax

```
 get_reconfig_modules [-regexp] [-nocase] [-filter arg]
[-of_objects args] [-quiet] [patterns]
```

Returns

list of reconfigurable module objects

Usage

Name	Optional	Default	Description
-regexp	yes		Patterns are full regular expressions
-nocase	yes		Perform case-insensitive matching (valid only when -regexp specified)
-filter	yes		Filter list with expression
-of_objects	yes		Get Reconfigurable Modules for these cells
-quiet	yes		Ignore command errors
patterns	yes	*	Match Reconfigurable Module names against patterns

Categories

Object, PartialReconfiguration

Description

This command returns a list of reconfigurable modules in the current project that match a specified search pattern. The default command returns a list of all reconfigurable modules in the project.

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error

Arguments

-regexp - Specifies that the search patterns are regular expressions.

-nocase - Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by get_reconfig_modules based on property values on the modules. You can find out what properties are on an object with the report_property or list_property commands.

The specific operators that can be used in the filter expression are ==, !=, and =~, as well as && and || between filter expressions.

-of_object *arg* - Get reconfigurable modules from the specified object.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

patterns - Match reconfigurable modules against the specified patterns. The default pattern is the wildcard '*' which returns all modules in the project.

Example

The following example returns a list of all reconfigurable modules in the project:

```
get_reconfig_modules
```

See Also

- create_reconfig_module
- list_property
- report_property

get_runs

Get a list of runs

Syntax

```
get_runs [-regexp] [-nocase] [-filter arg] [-quiet] [patterns]
```

Returns

list of run objects

Usage

Name	Optional	Default	Description
-regexp	yes		Patterns are full regular expressions
-nocase	yes		Perform case-insensitive matching (valid only when -regexp specified)
-filter	yes		Filter list with expression
-quiet	yes		Ignore command errors
patterns	yes	*	Match run names against patterns

Categories

Object, Project

Description

This command returns a list of synthesis and implementation runs in the current project that match a specified search pattern. The default command returns a list of all runs defined in the project.

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object, or string, to the list is not permitted and will result in a Tcl error

Arguments

-regexp - Specifies that the search patterns are regular expressions.

-nocase - Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by get_runs based on property values on the runs. You can find out what properties are on an object with the report_property command. Any property/value pair can be used as a filter. In the case of the "runs" object, "CONSTRSET", "IS_IMPLEMENTATION", "IS_SYNTHESIS", and "FLOW" are some of the properties that can be used to filter results.

Note: This argument is different from the filter command, which uses expressions to filter the results rather than property values.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

patterns - Match run names against the specified patterns. The default pattern is the wildcard '*' which returns all defined runs in the project. More than one pattern can be specified to find multiple runs based on different search criteria.

Examples

The following example returns a list of all runs defined in the current project:

```
get_runs -filter {IS_IMPLEMENTATION==1}
```

The following example returns a list of runs matching the pattern:

```
get_runs imp*
```

Note that if there are no runs matching the pattern the PlanAhead tool will return a warning indicating that no runs matched the specified pattern.

See Also

report_property

get_selected_objects

Get selected objects

Syntax

```
get_selected_objects [-primary] [-quiet]
```

Returns

list of selected objects

Usage

Name	Optional	Default	Description
-primary	yes		Do not include objects that were selected due to selection rules
-quiet	yes		Ignore command errors

Categories

Object, GUIControl

Description

Returns the objects currently selected by the **select_objects** command. Can return the primary selected object and any secondary selected objects as well.

Primary objects are directly selected, while secondary objects are selected by the PlanAhead tool based on the selection rules currently defined by the **Tools > Options** command. Refer to the *PlanAhead User Guide* (ug632) for more information on Setting Selection Rules.

Arguments

-primary - Indicates that only the primary selected object or objects should be returned; not secondary objects. As a default **get_selected_objects** will return all currently selected objects.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example reports the properties of all currently selected objects, both primary and secondary:

```
report_property [get_selected_objects]
```

See Also

select_objects

get_sites

Get a list of Sites

Syntax

```
 get_sites [-regexp] [-filter arg] [-range args]
[-of_objects args] [-quiet] [patterns]
```

Returns

list of site objects

Usage

Name	Optional	Default	Description
-regexp	yes		Patterns are regular expressions
-filter	yes		Filter list with expression
-range	yes		Match site names which fall into the range. Range is defined by exactly two site names.
-of_objects	yes		Get the sites of the objects passed in here.
-quiet	yes		Ignore command errors
patterns	yes	*	Match site names against patterns. Bonded sites will also match on package pin names.

Categories

XDC, Object

Description

This command returns a list of sites on the target device that match a specified search pattern. The default command returns a list of all sites on the target device.

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object to the list is not permitted and will result in a Tcl error

Arguments

-regexp - Specifies that the search patterns are regular expressions.

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by get_sites based on property values on the sites. You can find out what properties are on an object with the report_property command. In the case of the "site" object, "SITE_TYPE", "IS_OCCUPIED", "NUM_INPUTS", and "NUM_OUTPUTS" are some of the properties that can be used to filter results.

-range *arg* - Get all the sites that fall into a specified range. The range of sites must be specified with two site values, of the same SITE_TYPE, such as {Slice_X2Y12 Slice_X3Y15}. The SITE_TYPE of a site can be determined by the **report_property** command.

Note: Specifying a range with two different types will result in an error

-of_objects *arg* - Get sites from the specified object or objects. Only Tile, Bel, Pin, Package Pin, and I/O Banks are supported objects.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

patterns - Match sites against the specified patterns. The default pattern is the wildcard '*' which returns all sites on the target device.

Examples

The following example returns a list of all sites available on the target device:

```
get_sites
```

The following example returns the number of unoccupied sites on the device:

```
llength [get_sites -filter {IS_OCCUPIED==0}]
```

Note: If no sites match the pattern the PlanAhead tool will return a warning.

The following example gets all the sites on the device, and returns the unique SITE_TYPEs:

```
set sites [get_sites]
foreach x $sites {
   set prop [get_property SITE_TYPE $x]
   if { [lsearch -exact $type $prop] == -1 } { lappend type $prop } }
foreach y $type { puts "SITE_TYPE: $y" }
```

The following example returns the sites within the specified range of sites:

```
get_sites -range {Slice_X0Y0 Slice_X1Y1}
SLICE_X0Y0 SLICE_X0Y1 SLICE_X1Y0 SLICE_X1Y1
get_sites -range Slice_X0Y0 -range Slice_X1Y1
SLICE_X0Y0 SLICE_X0Y1 SLICE_X1Y0 SLICE_X1Y1
```

Note: The two methods of specifying the range of sites above return the same results.

See Also

- get_cells
- get_property
- list_property
- report_property

get_switching_activity

Get switching activity on specified objects

Syntax

```
get_switching_activity [-static_probability]
[-signal_rate] -object_list args [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-static_probability	yes		report static probability
-signal_rate	yes		report signal rate
-object_list	no		objects
-quiet	yes		Ignore command errors

Categories

XDC

get_timing_arcs

Get a list of timing arcs

Syntax

```
get_timing_arcs [-from args] [-to args] [-filter arg]
[-of_objects args] [-quiet]
```

Returns

list of timing arc objects

Usage

Name	Optional	Default	Description
-from	yes		List of pin or ports
-to	yes		List of pin or ports
-filter	yes		Filter list with expression
-of_objects	yes		Get timing arcs for these cells
-quiet	yes		Ignore command errors

Categories

SDC, XDC, Object

Description

This command returns a collection of timing arcs for custom reporting or other operations.

A filter expression can be used to filter the arcs according to specified properties.

The following properties are supported on timing arcs:

- delay_max_fall
- delay_max_rise
- delay_min_fall
- delay_min_rise
- from_pin
- is_annotated_fall_max
- is_annotated_fall_min
- is_annotated_rise_max
- is_annotated_rise_min
- is_cellarc
- is_disabled
- is_user_disabled
- mode
- object_class
- sdf_cond
- sense
- to_pin

Note: To improve memory and performance, the get_* commands return a container list of a single type of objects (i.e. cells, nets, pins, ports). You can add new objects to the list (using lappend for instance), but you can only add the same type of object that is currently in the list. Adding a different type of object to the list is not permitted and will result in a Tcl error

Arguments

-to *value* - (Optional) Specifies a list of to pins or to ports to get timing arcs for. All arcs forward of the specified pins or ports are considered.

-from *value* - (Optional) Specifies a list of from pins or from ports to get timing arcs for. All arcs backward of the specified pins or ports are considered.

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by get_cells based on property values on the cells. You can find out what properties are on an object with the report_property or list_property commands.

The specific operators that can be used in the filter expression are ==, !=, and =~, as well as && and || between filter patterns.

-of_objects *value* - (Optional) Specifies cells or nets to get timing arcs for. If a cell is specified, all cell_arcs of that cell are considered. If a net is specified, all net_arcs of that net are considered.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example calculates the timing arc From pin D to pin D of another flop:

```
get_timing_arcs -from infer_fifo.rd_addr_3/D -to infer_fifo.rd_addr_2/D
```

See Also

report_timing

group_path

Groups paths for cost function calculations

Syntax

```
group_path [-name arg] [-weight arg] [-default] [-from args]
[-to args] [-through args] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-name	yes		Name of the group
-weight	yes	1.0	Cost function weight: Range: 0 to 100 (Not Implemented)
-default	yes		Move path into the default group (Not Implemented)
-from	yes		Filter by paths starting at these path startpoints
-to	yes		Filter by paths terminating at these path endpoints
-through	yes		Consider paths through pins, cells or nets (Not Implemented)
-quiet	yes		Ignore command errors

Categories

SDC, XDC

Description

Groups a set of paths for cost function calculations, primarily for timing analysis. Timing paths can be specified generally as from a startpoint, or to an endpoint, or as from-through-to specific points.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-name <*arg*> - (Optional) Specifies the name of the path group. If the path group name already exists, the specified paths will be added to the existing group.

-default - (Optional) Moves the specified paths into the default group. The default group contains all paths not assigned to other groups. Use the **-default** argument to remove paths from a previously assigned path group.

Note: The **-name** and **-default** arguments are mutually exclusive

-weight *[arg]* - (Optional) Cost function weighting. Valid values range from 0 to 100, with 1.0 being the default value. The cost weighting is used raise the priority of the path group in cost function analysis.

-from *<args>* - (Optional) Include paths starting at the specified startpoints. The startpoints can be specified as pins or ports.

-to *<path_names>* - (Optional) Include all paths to the specified endpoints. Endpoints can be specified as ports or pins.

-through *<element_names>* - (Optional) Include paths routed through the specified pins, cells, or nets.

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example creates a group named fd_fd from the specified startpoint to the endpoint XLXI_2, and then reports timing on the specified group:

```
group_path -name fd_fd -from XLXI_1 -to XLXI_2
report_timing -group fd_fd
```

See Also

report_timing

help

Display help for one or more topics

Syntax

```
help [-category arg] [-short] [-quiet] [pattern]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-category	yes		Search for topics in the specified category
-short	yes		Display short help description
-quiet	yes		Ignore command errors
pattern	yes	*	Display help for topics that match the specified pattern

Categories

Project

Description

This command provides a list of Tcl command categories in the PlanAhead tool; a list of commands matching a specific pattern; brief or detailed help for a specific command.

The default **help** command without any arguments returns a list of Tcl command categories that can be further explored. Command categories are groups of command performing a specific function, like File I/O for instance.

Arguments

-category *arg* - Returns a list of commands grouped under the specified category.

-short - Provides an abbreviated help text for the specified command. The default of the tool is to return the detailed help for the specified command. Use this argument to keep it brief.

pattern - Returns a list of commands that match the specified search pattern. This form of help can be used to quickly locate a specific command from a group of commands.

command - Provides detailed information related to the specified command.

Examples

The following example returns the Tcl command categories in the PlanAhead tool:

```
help
```

The following example returns a list of all commands containing the specified search pattern:

```
help *file*
```

This list can be used to quickly locate a command for a specific purpose, such as remove_files or delete_files.

The following help command returns a detailed description of the remove_files command and its arguments:

```
help remove_files
```

Note: You can also use the **-short** option to get a brief description of the command.

highlight_objects

Highlight objects in different colors

Syntax

```
highlight_objects [-color_index arg] [-rgb args] [-color arg]
[-quiet] objects
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-color_index	yes		Color index
-rgb	yes		RGB color index list
-color	yes		Name of color
-quiet	yes		Ignore command errors
objects	no		Objects to highlight

Categories

GUIControl

Description

This command is used to highlight objects in the PlanAhead tool GUI mode. This command highlights the specified object or objects in a color as determined by one of the color options. Objects can be unhighlighted with the **unhighlight_objects** command.

Note: Only one color option should be used to specify the highlight color. However, if more than one color option is specified, the order of precedence used to define the color is -rgb, -color_index, and -color

Arguments

-color_index arg - Specify the color index to use for highlighting the selected object or objects. The color index is defined by the Highlight category of the Tools > Options > Themes command. Refer to the *PlanAhead User Guide* (ug632) for more information on setting themes.

-rgb args - Specify the color to use in the form of an RGB code specified as {R G B}. For instance, {255 255 0} specifies the color yellow.

-color arg - Specify the color to use for highlighting the specified object or objects. Supported highlight colors are: red, green, blue, magenta, yellow, cyan, and orange.

Note: White is the color used to display selected objects with the **select_objects** command

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

objects - Specifies one or more objects to be highlighted.

Example

The following example highlights the currently selected objects in the color red:

```
highlight_objects -color red [get_selected_objects]
```

See Also

- get_selected_objects
- unhighlight_objects

implement_debug_core

Implement ChipScope debug core

Syntax

```
implement_debug_core [-quiet] [cores ...]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
cores	yes		ChipScope debug core

Categories

ChipScope

Description

Use this command to implement the ChipScope debug core in the CORE Generator tool. The CORE Generator tool will be run once for any ILA debug cores specified, and run one more time for the ICON controller core if all cores are specified. The ILA core is the only core type currently supported by the create_debug_core command. The PlanAhead tool automatically adds an ICON controller core to contain and configure the ILA cores in the project.

The PlanAhead tool creates ChipScope ICON and ILA cores initially as black boxes. These cores must be implemented prior to running through place and route. After the core is created with create_debug_core, and while ports are being added and connected with **create_debug_port** and **connect_debug_port**, the content of the debug core is not defined or visible within the design.

ChipScope debug core implementation is automatic when you launch an implementation run using the **launch_runs** command. However, you can also use the **implement_debug_core** command to implement one or more of the cores in the CORE Generator tool without having to implement the whole design.

Arguments

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

cores - Specify one or more debug cores to implement.

Example

The following example implements all ChipScope debug cores in the current project:

```
implement_debug_core [get_debug_cores]
```

This results in both the ICON and ILA type cores being implemented in the CORE Generator tool for inclusion in the Netlist Design.

See Also

- connect_debug_port
- create_debug_core
- create_debug_port
- get_debug_cores
- launch_runs

import_as_run

Import an NCD and an optional TWX as a run

Syntax

```
import_as_run [-run arg] [-twx arg] [-pcf arg] [-quiet] ncd
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-run	yes		Import the results into this run
-twx	yes		TWX file to import
-pcf	yes		PCF file to import
-quiet	yes		Ignore command errors
ncd	no		Routed NCD file to import

Categories

Project

Description

Import an NCD and an optional TWX into an implementation run in the current project. This command is one of the steps involved in importing a previously placed and routed design from ISE into the PlanAhead tool.

Arguments

-run *arg* - Specifies the name of the implementation run to be imported into.

Note that this command does not create a run, but rather imports the required NCD file, and an optional TWX file if specified, into the specified run and sets its state to implemented.

-twx *arg* - Specifies the path and file name of an optional TRACE Timing file (.TWX) which can be imported along with the placement and routing data found in the .NCD file.

-pcf *arg* - Specifies a Physical Constraint File to load. Logical constraints in the NGD file are read by MAP. MAP uses some of the constraints to map the design and converts logical constraints to physical constraints and writes them to a Physical Constraints File (PCF).

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

ncd - Specify the path and name of the NCD file to import into the specified implementation run.

Examples

The following example creates a new empty RTL source project; changes the design_mode property to GateLvl, or netlist entry project; specifies the EDIF file representing the top-module of the design; and finally imports an NCD file into the implementation run which was initially created with the project.

```
create_project myProject C:/Data/myProject
set_property design_mode GateLvl [current_fileset]
set_property edif_top_file C:/Data/ise/drp_des/drp_demo.ngc [current_fileset]
import_as_run -run impl_1 C:/Data/ise/drp_des/drp_demo.ncd
```

Note that the implementation run is created along with a synthesis run when the project is first created. The **import_as_run** command cannot be used on that implementation run because the synthesis run has not yet been completed. However, when the design_mode property is next set to a GateLvl design, the synthesis run is removed as unnecessary, and only the implementation run is left.

The following example resets an implemented run to prepare it for use by the **import_as_run** command, and then imports an NCD file, a TWX file, and a PCF file from an ISE placed and routed design:

```
reset_run impl_1
import_as_run -run impl_1 -twx C:/Data/ise/drp_des/drp_demo.twx-pcf \ C:/Data/ise/drp_des/drp_
```

Note that if the **reset_run** command had not been used on the **impl_1** run, the following error would have been returned: ERROR: Run needs to be reset before importing into it.

See Also

- create_project
- reset_run
- set_property

import_files

Import files and/or directories into the active fileset

Syntax

```
 import_files [-fileset arg] [-force] [-norecurse] [-flat]
[-relative_to arg] [-quiet] [files ...]
```

Returns

A list of file objects that were imported

Usage

Name	Optional	Default	Description
-fileset	yes		Fileset name
-force	yes		Overwrite files of the same name in project directory
-norecurse	yes		Disables the default behavior of recursive directory searches
-flat	yes		Import the files into a flat directory structure
-relative_to	yes		Import the files with respect to the given relative directory
-quiet	yes		Ignore command errors
files	yes		Name of the files to import into fileset

Categories

Project, Simulation

Description

Imports one or more files or the source file contents of one or more directories to the specified fileset.

This command is different from the add_files command, which adds files by reference into the specified fileset. This command imports the files into the local project folders under `project.srcs\<fileset>\imports` and then adds the file to the specified fileset.

Arguments

-fileset *name* - Indicates which fileset the PlanAhead tool should add the specified source files to. If the specified fileset does not exist, the PlanAhead tool will return an error. If no fileset is specified the files will be added to the source fileset by default.

-force - Overwrite files of the same name in the local project directory and in the fileset.

-norecurse - This argument tells the PlanAhead tool not to recurse through subdirectories of any specified directories. Without this argument specified, the PlanAhead tool will also search through any subdirectories for additional source files that can be added to a project.

-flat - By default the PlanAhead tool preserves the directory structure of files as they are imported into the design. With this argument specified the PlanAhead tool imports all files into the imports folder without preserving their relative paths.

-relative_to *arg* - Import the files relative to the specified directory. This allows you to preserve the path to the imported files in the directory structure of the local project. The files will be imported to the imports folder with the path relative to the specified directory.

Note: The **relative_to** argument is ignored if the **-flat** argument is also specified. The **-flat** command eliminates the directory structure of the imported files.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

files - Provides a list of one or more file names or directory names to be added to the specified fileset. If a directory name is specified the PlanAhead tool will add all valid source files found in the directory, and in subdirectories of the directory.

Note: If the path is not specified as part of the file name, the PlanAhead tool will search for the specified file in the current working directory, or the directory from which the PlanAhead tool was launched.

Examples

The following example imports the top.ucf file into the constrs_1 constraint fileset.

```
import_files -fileset constrs_1 top.ucf
```

The following example imports the valid source files into the source fileset (`sources_1`) as a default since the **-fileset** argument is not specified. In addition, the **-norecurse** argument restricts the PlanAhead tool to looking only in the specified `\level1` directory and not searching any subdirectories. All valid source files will be imported into the `\imports` folder of the project because the **-flat** argument has been specified.

```
import_files C:/Data/FPGA_Design/level1 -norecurse -flat
```

Note: Without the **-flat** option the PlanAhead tool would have recreated the `\level1` directory inside of the `\imports` folder of the project.

The following example imports files into the source fileset (`sources_1`) because the **-fileset** argument is not specified. Valid source files are imported from the `\level1` directory, and all subdirectories, and the files will be written into the `\imports` folder of the project starting at the `\Data` directory due to the use of the **-relative_to** argument.

```
import_files C:/Data/FPGA_Design/level1 -relative_to C:/Data
```

See Also

add_files

import_ip

Import an IP file and add it to the fileset

Syntax

```
import_ip [-srcset arg] -file arg -name arg [-quiet]
```

Returns

list of file objects that were added

Usage

Name	Optional	Default	Description
-srcset	yes		Source set name
-file	no		Name of the IP file to be imported
-name	no		Name of new IP to be created
-quiet	yes		Ignore command errors

Categories

Project, COREGenerator

Description

Import an existing XCI or XCO file as IP into the current project.

Arguments

-file *arg* - Specifies the IP file to be imported into the current project. The IP must be in the form of an existing XCI file or XCO file. An XCI file is an IP-XACT format file that contains information about the IP parameterization. The XCO file is a CoreGen log file that records all the customization parameters used to create the IP core and the project options in effect when the core was generated. The XCI or XCO files are used by the PlanAhead tool to recreate the core in the current project.

-name *arg* - Specifies the name to assign to the IP object as it is added to the current source fileset.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example imports the 10gig ethernet core into the current project, and assigns it a name of IP_block1:

```
import_ip C:/Data/FPGA_Design/10gig_eth.xci -name IP_block1
```

The IP block is imported into the current source fileset in the current project.

See Also

- create_ip
- generate_ip

import_synplify

Imports the given Synplify project file

Syntax

```
import_synplify [-quiet] file
```

Returns

list of files object that were imported from the Synplify file

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
file	no		Name of the Synplify project file to be imported

Categories

Project

import_xise

Import XISE project file settings into the created project

Syntax

```
import_xise [-copy_sources] [-quiet] file
```

Returns

true

Usage

Name	Optional	Default	Description
-copy_sources	yes		Copy all ISE sources into the created project
-quiet	yes		Ignore command errors
file	no		Name of the XISE project file to be imported

Categories

Project

Description

Import an ISE project file (XISE) into the PlanAhead tool. This allows ISE projects to be quickly migrated into the PlanAhead tool for synthesis, simulation, and implementation. All the project source files, constraint files, simulation files, and run settings are imported from the ISE project and recreated in the current the PlanAhead tool project.

This command should be run on a new empty project. Since source files, constraints, and run settings are imported from the ISE project,any existing source files or constraints may be overwritten.

Arguments

-copy_sources - Specifies that source files in the ISE project should be copied to the local project directory structure rather than referenced from their current location. The default is to reference source files from their current location.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

file - Specifies the name of the ISE project file (.XISE) to be imported into the current project.

Example

The following example creates a new project called importISE, and then imports the ISE project file (`first_use.xise`) into the new project.

```
create_project importISE C:/Data/importISE import_xise \ C:/Data/FPGA_design/ise_designs/drp_
```

Note that this example does not specify the **-copy_sources** argument, so all source files in the ISE project will be added to the current project by reference.

See Also

create_project

import_xst

Imports the given XST project file

Syntax

```
import_xst [-quiet] file
```

Returns

list of files object that were imported from the XST file

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
file	no		Name of the XST project file to be imported

Categories

Project

Description

Import an XST synthesis project file into the PlanAhead tool, including the various source files used in the XST run to add to the current project.

Arguments

-copy_sources - Specifies that source files of the XST project should be copied to the local project directory structure rather than referenced from their current location. The default is to reference source files from their current location.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

file - Specifies the name of the XST project file to have the source files imported from.

Example

The following example creates a new project called `xst_test`, and imports the `drp_des.xst` file:

```
create_project xst_test C:/Data/FPGA_Design/xst_test import_xst \ C:/Data/ise_designs/drp_des
```

The source files specified in the specified XST project file are imported into the `xst_test` project.

See Also

create_project

launch_chipscope_analyzer

Launch ChipScope Analyzer tool for a run

Syntax

```
launch_chipscope_analyzer [-run arg] [-csproject arg] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-run	yes		Implemented run to launch ChipScope Analyzer with
-csproject	yes		ChipScope project
-quiet	yes		Ignore command errors

Categories

ToolLaunch, ChipScope

Description

Launch the ChipScope Pro Analyzer tool for the active run, or a specified Implemented Design run. You can setup a Netlist Design for use with ChipScope prior to implementation, using the **create_debug_core**, **create_debug_port**, and **connect_debug_port** commands.

The Implemented Design must also have a bitstream file generated by BitGen for **launch_chipscope_analyzer** to run. If BitGen has not been run, an error will be returned.

Note: It is not enough to use the **write_bitstream** command to create a bitstream file. You must follow the steps outlined below in the second example

Arguments

-run *arg* - Specify the run name to use when launching the ChipScope Pro Analyzer. The specified run must be implemented and have a bitstream (.bit) file generated. ChipScope will use the bitstream file and the `debug_nets.cdc` file from the specified run.

-csproject *arg* - Specify the name of the project to open in ChipScope Pro Analyzer. If you do not specify the project name, the default project name of `csdefaultproj.cpj` will be used. When you specify the project name, you should also specify the `.cpj` extension.

Note: The project is created in the `project/project.data/sources_1/cs` folder.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example launches ChipScope Pro Analyzer, specifying the implementation run to use and the name of the ChipScope project to create:

```
launch_chipscope_analyzer -run impl_3 -csproject impl_3_cs_project
```

The following example sets the **add_step Bitgen** property for the impl_4 run, launches the impl_4 run, and then launches the ChipScope Pro Analyzer on the specified run:

```
set_property add_step Bitgen [get_runs impl_4]
launch_runs impl_4 -jobs 2
launch_chipscope_analyzer -run impl_4
```

In this example the ChipScope project will be called `csdefaultproj.cpj`.

See Also

- connect_debug_port
- create_debug_core
- create_debug_port
- launch_runs
- set_property
- write_bitstream

launch_fpga_editor

Launch FPGAEditor tool for a Run

Syntax

```
launch_fpga_editor [-run arg] [-more_options arg] [-mapped_ncd]
[-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-run	yes		Implemented run to launch FED with
-more_options	yes		More command line options
-mapped_ncd	yes		Use Mapped NCD
-quiet	yes		Ignore command errors

Categories

ToolLaunch

Description

Launch the Xilinx FPGA Editor tool for the active run, or a specified run. When the design is open in the FPGA Editor, you can place critical elements of the design and examine critical timing pathways. You can also cross-probe between the PlanAhead tool and the FPGA Editor using the **crossprobe_fed** command to quickly locate elements of your design in the two editors.

Arguments

-run *arg* - Specify the run name to use when launching the FPGA Editor. The specified run must have a mapped NCD or routed NCD file to launch the FPGA Editor.

-more_options *arg* - Specify additional options to use when invoking the FPGA Editor software. For more information, see the online Help provided with FPGA Editor.

-mapped_ncd - Use the mapped NCD file as input for FPGA Editor. This allows you to view the NCD file output from MAP, without the placement and routing data from PAR. You can use this to perform some preplacement and routing of critical components if necessary.

Note: By default the **launch_fpga_editor** command loads the *design*_routed.ncd file, unless this option is specified.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example launches FPGA Editor, specifying the implementation run to use and to open the mapped NCD instead of the placed and routed NCD:

```
launch_fpga_editor -run impl_4 -mapped_ncd
```

See Also

crossprobe_fed

launch_impact

Launch iMPACT configuration tool for a run

Syntax

```
launch_impact [-run arg] [-ipf arg] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-run	yes		Implemented run to launch iMPACT with
-ipf	yes		Project for iMPACT
-quiet	yes		Ignore command errors

Categories

ToolLaunch

launch_isim

Launch simulation using ISim simulator

Syntax

```
launch_isim [-simset arg] [-mode arg] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-simset	yes		Name of the simulation fileset
-mode	yes	behavioral	Simulation mode. Values: behavioral, timing
-quiet	yes		Ignore command errors

Categories

ToolLaunch, Simulation

launch_runs

Launch a set of runs

Syntax

```
launch_runs [-jobs arg] [-scripts_only] [-all_placement]
[-dir arg] [-to_step arg] [-host args] [-remote_cmd arg]
[-email_to args] [-email_all] [-pre_launch_script arg]
[-post_launch_script arg] [-force] [-quiet] runs ...
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-jobs	yes	1	Number of jobs
-scripts_only	yes		Only generate scripts
-all_placement	yes		Export all fixed and non-fixed placement to ISE (by default only fixed placement will be exported)
-dir	yes		Launch directory
-to_step	yes		Last Step to run
-host	yes		Launch on specified remote host with a specified number of jobs. Example: -host {machine1 2} -host {machine2 4}
-remote_cmd	yes	ssh -q -o BatchMode=yes	Command to log in to remote hosts
-email_to	yes		List of email addresses to notify when jobs complete
-email_all	yes		Send email after each job completes
-pre_launch_script	yes		Script to run before launching each job
-post_launch_script	yes		Script to run after each job completes
-force	yes		Run the command, even if there are pending constraint changes, which will be lost (in a Partial Reconfig design)
-quiet	yes		Ignore command errors
runs	no		Runs to launch

Categories

Project

Description

Launches synthesis and implementation runs.

The run must be defined using the **create_run** command, and the attributes of the run must be previously configured using the **set_property** command. Both synthesis and implementation runs can be specified in the same **launch_runs** command. However, to launch an implementation run, the parent synthesis run must already be complete.

Arguments

-jobs *arg* - Te number of parallel jobs to run on the local host. Note that the number of jobs for a remote host is specified as part of the **-host** argument. You do not need to specify both **-jobs** and **-host**.

-scripts_only - Generate a script called `runme.bat` for each specified run so you can queue the runs to be launched at a later time.

-all_placement - Export all user-assigned (fixed) placements as well as auto-assigned (unfixed) placements to ISE MAP and PAR tools for implementation. As a default, the PlanAhead tool will export only the fixed or user-assigned placement for implementation.

-dir *arg* - The directory in which to write run results. A separate folder for each run is created under the specified directory. As a default, results of each run are placed into a separate folder under the *project*`.runs` directory.

-to_step *arg* - Launch the run through the specified step in the process, and then stop. For instance, for implementation runs, run through the MAP step, and then stop. This will allow you to look at specific stages of a run without completing the entire run. Valid values are: NGDBuild, MAP, PAR, TRCE, XDL.

-next_step - Continue a prior run from the step it was stopped at. This option can be used to complete a run previously launched with the **-to_step** argument.

Note: The **-to_step** and **-next_step** arguments may not be specified together; are only valid for ISE implementation runs; and are ignored when launching multiple runs

-host *args* - This command is supported on the Linux platform only. Launch on the named remote host with a specified number of jobs. The argument is in the form of {*hostname jobs*}, for example: -host {machine1 2}. If the **-host** argument is not specified, the runs will be launched from the local host.

-remote_cmd *arg* - Specifies the command to use to login to the remote host to launch jobs. The default remote command is "ssh -q -o BatchMode=yes".

-email_to *args* - Email addresses to send a notification when the runs have completed processing.

-email_all - Send a separate Email for each run as it completes.

-pre_launch_script *arg* - Specifies a script to run before launching each job.

-post_launch_script *arg* - Specifies a script to run after completion of all jobs.

-force - This argument applies only to Partial Reconfiguration projects. Launch the run regardless of any pending constraint changes for Partial Reconfiguration designs. Note that any pending constraint changes will be lost to the specified runs.

-quiet - Execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

runs - Specify the names of synthesis and/or implementation runs to launch.

Examples

The following command launches three different synthesis runs with two parallel jobs:

```
launch_runs synth_1 synth_2 synth_4 -jobs 2
```

Note: The results for each run will be written to a separate folder `synth_1`, `synth_2`, and `synth_4` inside of the *project*`.runs` directory.

The following example creates a results directory to write run results. In this case a separate folder named `impl_3`, `impl_4`, and `synth_3` will be written to the specified directory. In addition, the **-scripts_only** argument tells PlanAhead to write `runme.bat` scripts to each of these folders but not to launch the runs at this time.

```
launch_runs impl_3 impl_4 synth_3 -dir C:/Data/FPGA_Design/results -scripts_only
```

See Also

- create_run
- get_runs
- set_property

launch_xpa

Launch XPower Analyzer tool

Syntax

```
launch_xpa [-run arg] [-more_options arg] [-mapped_ncd]
[-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-run	yes		Implemented run to launch XPA with
-more_options	yes		More command line options
-mapped_ncd	yes		Use Mapped NCD
-quiet	yes		Ignore command errors

Categories

ToolLaunch

list_param

Get all parameter names

Syntax

```
list_param [-quiet]
```

Returns

list

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors

Categories

PropertyAndParameter

Description

This command returns a list of the user-definable configuration parameters within the PlanAhead tool. These parameters configure a variety of settings and behaviors of the tool. For more information on a specific parameter refer to the **report_param** command, which returns a description of the parameter as well as its current value.

Arguments

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example returns a list of all user-definable parameters within the PlanAhead tool:

```
list_param
```

See Also

- get_param
- report_param
- reset_param
- set_param

list_property

List properties of object

Syntax

```
list_property [-class arg] [-quiet] [object]
```

Returns

list of property names

Usage

Name	Optional	Default	Description
-class	yes		Object type to query for properties. Ignored if object is specified.
-quiet	yes		Ignore command errors
object	yes		Object to query for properties

Categories

Object, PropertyAndParameter

Description

This command takes an object or a class of objects as input and returns a list of all properties on the object or class.

Note: report_property also returns a list of properties on an object, but includes the property type and property value.

Arguments

-class *arg* - Specifies a class of object to list the properties of. When both **-class** and *object* are specified, the properties of the specific object are returned.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

object - Specifies one object to report properties.

Note: If you specify multiple objects the PlanAhead tool will return an error.

Example

The following example returns all properties of the specified object:

```
list_property [get_cells cpuEngine]
```

See Also

- create_property
- get_cells
- get_property
- list_property_value
- report_property
- reset_property
- set_property

list_property_value

List legal property values of object

Syntax

```
list_property_value [-class arg] [-quiet] name [object]
```

Returns

list of property values

Usage

Name	Optional	Default	Description
-class	yes		Object type to query for legal property values. Ignored if object is specified.
-quiet	yes		Ignore command errors
name	no		Name of property whose legal values is to be retrieved
object	yes		Object to query for legal properties values

Categories

Object, PropertyAndParameter

Description

Returns a list of legal values for an enumerated type property of either a class of objects or a specific object.

Note: The command cannot be used to return legal values for properties other than enum properties. The report_property command will return the type of property to help you identify enum properties.

Arguments

-class *arg* - Specifies the class of object to query. The class of object can be used in place of an actual object.

name - Specifies the name of the property to be queried. Only properties with an enumerated value, or a predefined value set, can be queried with this command. All legal values of the specified property will be returned.

object - Specifies an object to query. An actual object can be used in place of the -class argument to specify the type of object to query.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example returns the list of legal values for the KEEP_HIERARCHY property from cell objects:

```
list_property_value KEEP_HIERARCHY -class cell
```

The following example returns the same result, but uses an actual cell object in place of the general cell class:

```
list_property_value KEEP_HIERARCHY [get_cells cpuEngine]
```

See Also

- create_property
- get_cells
- get_property
- list_property
- report_property
- reset_property
- set_property

load_reconfig_modules

Load specific Reconfigurable Modules or all modules from a given run.

Syntax

```
load_reconfig_modules [-force] [-run arg] [-quiet]
[reconfig_modules ]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-force	yes		Run the command, even if there are pending constraint changes, which will be lost
-run	yes		Run to load Reconfigurable Modules from
-quiet	yes		Ignore command errors
reconfig_modules	yes		Reconfigurable Module(s) to load

Categories

PartialReconfiguration

make_diff_pair_ports

Make differential pair for 2 ports

Syntax

```
make_diff_pair_ports [-quiet] ports ...
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
ports	no		Ports to join

Categories

PinPlanning

mark_objects

Mark objects in GUI

Syntax

```
mark_objects [-rgb args] [-color arg] [-quiet] objects
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-rgb	yes		RGB color index list
-color	yes		Name of color
-quiet	yes		Ignore command errors
objects	no		Objects to mark

Categories

GUIControl

Description

This command is used to mark specified objects in the PlanAhead tool GUI mode. This command places an iconic mark to aid in the location of the specified object or objects. The mark is displayed in a color as determined by one of the color options.

Objects can be unmarked with the **unmark_objects** command.

Note: Only one color option should be used to specify the color. However, if both color options are specified, -rgb takes precedence over -color

Arguments

-rgb args - Specify the color to use in the form of an RGB code specified as {R G B}. For instance, {255 255 0} specifies the color yellow, while {0 255 0} specifies green.

-color arg - Specify the color to use for marking the specified object or objects. Supported colors are: red, green, blue, magenta, yellow, cyan, and orange.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

objects - Specifies one or more objects to be marked.

Example

The following example adds a red icon to mark the currently selected objects:

```
mark_objects -color red [get_selected_objects]
```

See Also

- get_selected_objects
- unmark_objects

open_impl_design

Open an implementation design

Syntax

```
open_impl_design [-quiet] [run]
```

Returns

design object

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
run	yes		Run to open

Categories

Project

Description

Opens an Implemented Design.

Arguments

-quiet - This option tells the PlanAhead software to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

run - Specifies the run name of the implemented design to open. You must first implement the specified run before opening the Implemented Design. If you attempt to open a run that has not been implemented the PlanAhead software will return an error.

Example

The following opens the Implemented Design for impl_1:

```
open_impl_design impl_1
```

See Also

- config_run
- launch_runs
- open_netlist_design
- open_rtl_design

open_io_design

Open an IO design

Syntax

```
open_io_design [-name arg] [-part arg] [-constrset arg]
[-quiet]
```

Returns

design object

Usage

Name	Optional	Default	Description
-name	yes		Design name
-part	yes		Target part
-constrset	yes		Constraint fileset to use
-quiet	yes		Ignore command errors

Categories

Project

Description

Opens a new or existing I/O Pin Planning design.

Note: The design_mode property for the current source fileset must be defined as PinPlanning in order to open an I/O design. The PlanAhead tool will return the following error if that is not the case: ERROR: The design mode of 'sources_1' must be PinPlanning

Arguments

-name *arg* - Specifies the name of a new or existing I/O Pin Planning design.

-part *arg* - Specifies the Xilinx device to use when creating a new design. If the part is not specified the default part will be used.

-constrset *arg* - Specifies the name of the constraint fileset to use when opening an I/O design.

Note: The -constrset argument must refer to a constraint fileset that exists. It cannot be used to create a new fileset. Use create_fileset for that purpose.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following creates a new I/O design called myIO:

```
open_io_design -name myIO
```

Note: The default source set, constraint set, and part will be used in this case.

The following example opens an existing I/O design called myIO, and specifies the constraint set to be used:

```
open_io_design -name myIO -constrset topCon
```

See Also

create_project

open_netlist_design

Open a synthesis or netlist design

Syntax

```
 open_netlist_design [-name arg] [-run arg] [-part arg]
[-constrset arg] [-top arg] [-block] [-quiet]
```

Returns

design object

Usage

Name	Optional	Default	Description
-name	yes		Design name
-run	yes		Run to open into the netlist design
-part	yes		Target part
-constrset	yes		Constraint fileset to use
-top	yes		Specify the top module name when the structural netlist is Verilog
-block	yes		Open a block-level design
-quiet	yes		Ignore command errors

Categories

Project

Description

Opens a new or existing Netlist design.

The design_mode property for the current source fileset must be defined as GateLvl in order to open a Netlist design. the PlanAhead software will return the following error if that is not the case: ERROR: The design mode of 'sources_1' must be GateLvl

Use the **set_property** command to set the design_mode property.

Arguments

-name *arg* - Specifies the name of a new or existing Netlist design.

-run *arg* - Specifies the implementation run to open with the Netlist design.

-part *arg* - Specifies the Xilinx device to use when creating a new design. If the part is not specified the default part will be used.

-constrset *arg* - Specifies the name of the constraint fileset to use when opening the design.

Note: The -constrset argument must refer to a constraint fileset that exists. It cannot be used to create a new fileset. Use create_fileset for that purpose.

-top *arg* - Specify the top module of the design hierarchy when the Netlist design is Verilog.

-quiet - This option tells the PlanAhead software to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following creates a new Netlist design called Net1:

```
open_netlist_design -name Net1
```

Note: The default source set, constraint set, and part will be used in this case.

The following example opens an existing Netlist design called Net1, and specifies the constraint set to be used:

```
open_netlist_design -name Net1 -constrset con1
```

Note: The default source set and part will be used in this case.

See Also

- config_run
- launch_runs
- open_impl_design
- open_rtl_design

open_project

Open a PlanAhead project file (.ppr)

Syntax

```
open_project [-read_only] [-quiet] file
```

Returns

opened project object

Usage

Name	Optional	Default	Description
-read_only	yes		Open the project in read-only mode
-quiet	yes		Ignore command errors
file	no		Project file to be read

Categories

Project

Description

Opens a PlanAhead project file (.ppr).

This command opens a project file for editing the design source files and hierarchy, for performing I/O pin planning and floorplanning, and to synthesize and implement the device.

Arguments

-read_only - Specifies that the project should be opened in read only mode and does not support saving any modifications of the project. However, you can also use the **save_project_as** command to save a read_only project to a new project which will be editable.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

file - Specifies the PlanAhead project file to open. The file specified must include both the path to the file and the `.ppr` file extension.

Example

The following example opens the project named `my_project1` located in the Designs directory.

```
open_project C:/Designs/my_project1.ppr
```

Note: The project must be specified with the `.ppr` extension for the PlanAhead tool to recognize it as a project file. The path to the file must be specified along with the project file name or the PlanAhead tool will return an error that it cannot find the specified file.

See Also

- create_project
- current_project

open_rtl_design

Open an rtl design

Syntax

```
 open_rtl_design [-name arg] [-part arg] [-constrset arg]
[-top arg] [-quiet]
```

Returns

design object

Usage

Name	Optional	Default	Description
-name	yes		Design name
-part	yes		Target part
-constrset	yes		Constraint fileset to use
-top	yes		Specify the top module name
-quiet	yes		Ignore command errors

Categories

Project

Description

Opens a new or existing RTL source design. Note that Verilog or VHDL source files must be added to the source fileset, and a top module identified, before opening an RTL design.

Arguments

-name *arg* - Specifies the name of a new or existing RTL design.

-part *arg* - Specifies the Xilinx device to use when creating a new design. If the part is not specified the default part will be used.

-constrset *arg* - Specifies the name of the constraint fileset to use when opening the design.

Note: The -constrset argument must refer to a constraint fileset that exists. It cannot be used to create a new fileset. Use create_fileset for that purpose.

-quiet - This option tells the PlanAhead software to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following creates a new RTL design called RTL1:

```
open_rtl_design -name RTL1
```

Note: The default source set, constraint set, and part will be used in this case.

The following example opens an existing RTL design called RTL1, and specifies the constraint set to be used:

```
open_rtl_design -name RTL1 -constrset top
```

Note: The default source set and part will be used in this case.

See Also

- config_run
- launch_runs
- open_impl_design
- open_netlist_design

place_cell

Move or place one or more instances to new locations. Sites and cells are required to be listed in the right order and there should be same number of sites as number of cells.

Syntax

```
place_cell [-quiet] cell_site_list ...
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
cell_site_list	no		a list of cells and sites in the interleaved order

Categories

Floorplan

place_pblocks

Run the pblocks Placer

Syntax

```
 place_pblocks [-effort arg] [-utilization arg]
[-quiet] pblocks ...
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-effort	yes	HIGH	Placer effort level (per pblock) Values: LOW, MEDIUM, HIGH
-utilization	yes		Placer utilization (per pblock)
-quiet	yes		Ignore command errors
pblocks	no		List of pblocks to place

Categories

Floorplan

Description

Place Pblocks onto the fabric of the FPGA. Pblocks must be created using the create_pblock command, and should be populated with assigned logic using the add_cells_to_pblock command.

Note: An empty Pblock will be placed as directed, but results in a Pblock covering a single CLB tile (two slices).

Arguments

-effort *arg* - Specifies the level of effort that the Pblock placer should use in placing each Pblock onto the fabric. The legal values are LOW, MEDIUM, HIGH, with the default being HIGH when this option is not specified.

-utilization *arg* - Specifies percentage of device resources that should be consumed by the logic elements assigned to a Pblock when it is placed onto the FPGA. For instance, a utilization rate of 50% means that half of the resources should be allocated to the logic in the Pblock, and half should be left for other design elements to be intermingled. A high utilization rate makes the Pblock smaller but more difficult to place, while a smaller utilization makes the Pblock larger.

Note: Pblock utilization is post-synthesis estimation. Actual results may be different, and may require you to resize the Pblock using the **resize_pblock** command

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

pblocks - Specify one or more Pblocks to be placed onto the fabric of the FPGA.

Example

The following example places the specified Pblocks with a utilization of 75%:

```
place_pblocks -effort LOW -utilization 75 block1 block2 block3 block4 block5
```

See Also

- add_cells_to_pblock
- create_pblock
- resize_pblock

place_ports

Automatically place a set of ports

Syntax

```
 place_ports [-skip_unconnected_ports] [-check_only] [-quiet]
[ports ...]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-skip_unconnected_ports	yes		Do not place unconnected ports
-check_only	yes		Only check IO/Clock placement DRCs
-quiet	yes		Ignore command errors
ports	yes		Ports to place (if omitted, all ports will be placed)

Categories

PinPlanning

promote_run

Promote previously implemented Partitions to make them available to import and reuse in this or other runs

Syntax

```
promote_run [-partition_names args] [-promote_dir arg]
[-description arg] [-no_state_update] [-quiet] run
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-partition_names	yes		List of Partitions to be promoted
-promote_dir	yes		Promote path
-description	yes		Promote description
-no_state_update	yes		Do not automatically update the state of promoted Partitions to Import
-quiet	yes		Ignore command errors
run	no		Implemented run to be promoted

Categories

PartialReconfiguration, Partition

read_chipscope_cdc

Import ChipScope Core Inserter CDC file

Syntax

```
read_chipscope_cdc [-quiet] file
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
file	no		ChipScope CDC file name

Categories

FileIO, ChipScope

Description

Read a ChipScope Definition and Connection (CDC) file to associate with a Netlist Design in the current project. This file stores information about core parameters, and core settings for ChipScope ILA debug cores, and can also be used as input to the ChipScope Pro Analyzer to import signal names.

The Chipscope CDC file can come from an ISE project or from another the PlanAhead tool project through the write_chipscope_cdc command.

If certain parameters of the CDC file are not acceptable to the current project then those parameters will not be imported. For instance, if signals specified for connection to ports do not exist in the current netlist, those signals will be ignored.

Arguments

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

file - Specify the path and filename of the CDC file to read into the PlanAhead tool. The CDC file defines the debug core to insert, the signals to probe, and the clock domains for those signals.

Example

The following example reads the specified CDC file:

```
read_chipscope_cdc C:/Data/FPGA_Design/bft.cdc
```

See Also

- create_debug_core
- get_debug_cores
- write_chipscope_cdc

read_csv

Import package pin and port placement information

Syntax

```
read_csv [-quiet] file
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
file	no		Pin Planning CSV file

Categories

FileIO

Description

Import package pin and port placement information from a comma separated value (CSV) file.

The CSV file can only be imported into an IO Pin Planning project. In all other projects the pin definitions are imported with the source design data.

The specific format and requirements of the CSV file are described in the *PlanAhead Users Guide* (ug632.pdf).

Arguments

file - Specifies the file name of the CSV file to be imported.

Note: If the path is not specified as part of the file name, the PlanAhead tool will search for the specified file in the current working directory, or the directory from which the PlanAhead tool was launched.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example imports a CSV file into an open project:

```
read_csv C/Data/pinList.csv
```

The following example sets up a new IO Pin Planning project, and then imports the specified CSV file into it:

```
create_project myPinPlan C:/Data/myPinPlan -part xc7v285tffg1157-1
set_property design_mode PinPlanning [current_fileset]
open_io_design -name io_1
read_csv C:/Data/import.csv
```

Note: The design_mode property on the source fileset is what determines the nature of the project.

See Also

- create_project
- open_io_design
- set_property
- write_csv

read_edif

Read one or more EDIF files

Syntax

```
read_edif [-quiet] files
```

Returns

list of file objects that were added

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
files	no		EDIF file name(s)

Categories

FileIO

Description

Import an EDIF netlist file into the Design Source fileset of the current project.

Arguments

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

files - Specifies the file name of the EDIF file to be imported.

Note: If the path is not specified as part of the file name, the PlanAhead tool will search for the specified file in the current working directory, or the directory from which the PlanAhead tool was launched.

Example

The following example imports an EDIF file into the open project:

```
read_edif C/Data/bft_top.edf
```

See Also

write_edif

read_pxml

Import Partition definitions from a PXML file.

Syntax

```
read_pxml [-quiet] file
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
file	no		PXML file name

Categories

FileIO

Description

Import a partition XML file into the current design. The PXML file contains partition information related to hierarchical design. A PXML file is created by the PlanAhead tool during synthesis or implementation. You can also create one by hand or by using an XML template provided in the PlanAhead tool installation directory. Refer to the *Hierarchical Design Methodology Guide* (ug748) for more information on partitioning designs and creating a PXML file.

A partition must be defined to implement, or to import in the RTL design in order to be properly handled in synthesis and implementation. Therefore read_pxml must be used on an open RTL design.

Arguments

file - Specifies the name of the PXML file. The file must be named xpartition.pxml, or the PlanAhead tool will return an error when trying to read the file.

Note: If the path is not specified as part of the file name, the PlanAhead tool will search for the specified file in the current working directory, or the directory from which the PlanAhead tool was launched.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example reads the specified PXML file for partition information related to the hierarchy of the design:

```
read_pxml C:/Data/FPGA_Design/pxmlTest.pxml
```

See Also

config_partition

read_twx

Read timing results from Trace STA tool

Syntax

```
read_twx [-cell arg] [-pblock arg] [-quiet] name file
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-cell	yes		Interpret names in the report file as relative to the specified cell
-pblock	yes		Interpret names in the report file as relative to the specified pblock
-quiet	yes		Ignore command errors
name	no		Name for the set of results
file	no		Name of the Trace import file

Categories

FileIO

Description

Import timing results in the TWX format timing report files generated by the Xilinx Timing Reporter And Circuit Evaluator (TRACE) tool. The TWX file can be imported at the top-level, which is the default, or at a specific cell-level or relative to a specific Pblock.

After the TWX files are imported, the timing results display in the Timing Results view in GUI mode.

Arguments

-cell *arg* - Specify the name of a hierarchical cell in the current design to import the TWX file into. The timing paths will be applied to the specified cell.

-pblock *arg* - Specify the name of a Pblock in the current design. The timing paths will be imported relative to the specified block.

name - Specifies the name of the Timing Results view to create when importing the timing paths in the TWX file.

Note: Both *name* and *file* are required positional arguments. The *name* argument must be provided first.

file - Specifies the file name of the TWX file to be imported.

Note: If the path is not specified as part of the file name, the PlanAhead tool will search for the specified file in the current working directory, or the directory from which the PlanAhead tool was launched.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example reads the specified TWX file into the top-level of the design:

```
read_twx C:/Data/timing_files/bft.twx
```

See Also

report_timing

read_ucf

Import physical constraints from a file.

Syntax

```
read_ucf [-cell arg] [-quiet] file
```

Returns

list of added files

Usage

Name	Optional	Default	Description
-cell	yes		import constraints for this cell
-quiet	yes		Ignore command errors
file	no		file name

Categories

FileIO

Description

Import physical constraints from a user constraint file (UCF). The UCF can be imported at the top-level, which is the default, or at a specific cell-level. When imported at the top-level, the specified UCF file is added to the active constraint fileset.

This command is similar to the add_files command in that the UCF file is added by reference rather than imported into the local project directory.

Note: Constraints from the UCF file will overwrite any current constraints of the same name. Therefore, exercise some caution when reading a UCF file to be sure you will not overwrite important constraints.

Arguments

file - Specifies the file name of the UCF file to be imported.

Note: If the path is not specified as part of the file name, the PlanAhead tool will search for the specified file in the current working directory, or the directory from which the PlanAhead tool was launched.

-cell *arg* - Specify the name of a hierarchical cell in the current design to import the UCF file into. The constraints will be applied to the specified block, and the imported UCF file will not be added to the active constraint fileset.

Note: A design must be open when specifying the -cell option.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example reads the specified UCF file into the top-level of the design:

```
read_ucf C:/Data/FPGA_Design/top1.ucf
```

See Also

- add_files
- write_ucf
- save_design

read_verilog

Read one or more Verilog files

Syntax

```
read_verilog [-library arg] [-sv] [-quiet] files ...
```

Returns

list of file objects that were added

Usage

Name	Optional	Default	Description
-library	yes	work	library name
-sv	yes		Enable system verilog compilation
-quiet	yes		Ignore command errors
files	no		Verilog file name(s)

Categories

FileIO

Description

Read Verilog or SystemVerilog source files. This command is very similar to the add_files command. The Verilog file is added to the source fileset as it is read. If the **-library** argument is specified, the file is added with the Library property defined appropriately.

Because SystemVerilog is a superset of the Verilog language, the **read_verilog** command can read both file types. However, for SystemVerilog files, the **-sv** option needs to be specified for **read_verilog** to enable compilation in the SystemVerilog mode. In this mode, the PlanAhead tool recognizes and honors the SystemVerilog keywords and constructs.

In a single PlanAhead project, you can have a mixture of both Verilog files (.v files), and SystemVerilog files (.sv files), as well as VHDL (using **read_vhdl**). When the PlanAhead tool compiles these files for synthesis, it creates separate "compilation units" for each file type. All files of the same type are compiled together.

Arguments

-library *arg* - Specify the library the Verilog file should reference. The default Verilog library is work.

-sv - Use this option to specify that the files should be read in as a SystemVerilog compilation group.

Note: Since Verilog is a subset of SystemVerilog, unless a Verilog source has user-defined names that collide with reserved SystemVerilog keywords, reading Verilog files with the -sv switch enables SystemVerilog compilation mode for those files. However, adding a SystemVerilog file in a Verilog compilation unit (without **-sv**) will not work

files - Specifies the name of one or more Verilog files to be read.

Note: If the path is not specified as part of the file name, the PlanAhead tool will search for the specified file in the current working directory, or the directory from which the PlanAhead tool was launched.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example reads the specified Verilog file and adds it to the source fileset:

```
read_verilog C:/Data/FPGA_Design/new_module.v
```

The following example creates two compilation units, one for SystemVerilog files and one for Verilog files:

```
read_verilog -sv { file1.sv file2.sv file3.sv }
read_verilog { file1.v file2.v file3.v}
```

See Also

- add_files
- read_vhdl

read_vhdl

Read one or more VHDL files

Syntax

```
read_vhdl [-library arg] [-quiet] files
```

Returns

list of file objects that were added

Usage

Name	Optional	Default	Description
-library	yes	work	vhdl library
-quiet	yes		Ignore command errors
files	no		VHDL file name(s)

Categories

FileIO

Description

Read a VHDL source file. This command is very similar to the add_files command. The VHDL file is added to the source fileset as it is read. If the -library argument is specified, the file is added with the Library property defined appropriately.

Arguments

-library *arg* - Specify the library the VHDL file should reference. The default VHDL library is work.

file - Specifies the file name of the VHDL file to be read.

Note: If the path is not specified as part of the file name, the PlanAhead tool will search for the specified file in the current working directory, or the directory from which the PlanAhead tool was launched.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example reads the specified VHDL file and adds it to the source fileset:

```
read_vhdl C:/Data/FPGA_Design/new_module.vhdl
```

See Also

add_files

read_xdl

Import placement information from a file

Syntax

```
read_xdl [-pblock arg] [-cell arg] [-quiet] file
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-pblock	yes		Import placement for this pblock
-cell	yes		Import placement for this cell
-quiet	yes		Ignore command errors
file	no		Placement file name

Categories

FileIO

Description

The PlanAhead tool can import ISE placement results using XDL format data. XDL data is created automatically when Implementation runs are launched. You can also create an XDL format file from an NCD file using the XDL command-line tool.

You can create XDL files and import placement for the entire design, for individual modules, or relative to specific Pblocks.

Arguments

-pblock *arg* - Specify the name of a Pblock in the current design. The data in the XDL file will be imported relative to the specified block.

-cell *arg* - Specify the name of a hierarchical cell in the current design to import the XDL file into. The data in the XDL file will be imported relative to the specified cell.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

file - Specifies the file name of the XDL file to be imported.

Note: If the path is not specified as part of the file name, the PlanAhead tool will search for the specified file in the current working directory, or the directory from which the PlanAhead tool was launched.

Example

The following example reads the specified XDL file into the top-level of the design:

```
read_xdl C:/Data/FPGA_Designs/bft.xdl
```

redo

Re-do previous command

Syntax

```
redo [-list] [-quiet]
```

Returns

with -list, the list of redoable tasks

Usage

Name	Optional	Default	Description
-list	yes		Show a list of redoable tasks
-quiet	yes		Ignore command errors

Categories

GUIControl

Description

This command will redo a command that has been previously undone. This command can be used repeatedly to redo a series of commands.

If a command group has been created using the **startgroup** and **endgroup** commands, the redo command will redo the group of commands as a sequence.

Arguments

-list - Reports the list of commands that can be redone. When you use the **undo** command, the PlanAhead tool will step backward through a list of commands. The **redo** command can then be used to redo those commands.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example returns a list of commands that can be redone:

```
redo -list
```

See Also

- endgroup
- startgroup
- undo

refresh_design

Refresh the current design

Syntax

```
refresh_design [-part arg] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-part	yes		Target part
-quiet	yes		Ignore command errors

Categories

Project

Description

Reloads the current design from the project data on the hard drive. This overwrites the in-memory view of the design to undo any recent design changes.

Arguments

-part *arg* - Specifies that the design should be reloaded with the specified part as the new target part for the design. This overrides the constraint file part specified in the project data on the hard drive.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following command reloads the current design from the project data on hard disk. This will overwrite the unsaved changes of the design which are in memory.

```
refresh_design
```

You can use the command to undo a series of changes to the design and revert to the previously saved design.

The following example refreshes the current design using the specified V6 part as the target device. The second command is required to make the selected part the target device for the active implementation run.

```
refresh_design -part xc6vcx75tff784-1
set_property part xc6vcx75tff784-1 [get_runs impl_6]
```

Note: The second command is not required if the target part is not changed.

See Also

set_property

reimport_files

Reimport files when they are found out-of-date

Syntax

```
reimport_files [-force] [-quiet] [files ...]
```

Returns

list of file objects that were imported

Usage

Name	Optional	Default	Description
-force	yes		Force a reimport to happen even when the local files may be newer
-quiet	yes		Ignore command errors
files	yes		List of files to reimport. If no files are specified, all files in the project that are out-of-date, will be reimported

Categories

Project

Description

Causes the PlanAhead tool to reimport project files. This updates the local project files from the original referenced source files.

Arguments

-force - Forces the PlanAhead tool to reimport files even when the local project files may be newer than their referenced source files.

-quiet - This option tells PlanAhead to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

files - List of files to reimport. If no files are specified, all files in the project that are out-of-date, will be reimported. If -force is used and no files are specified, all files in the project will be reimported.

Examples

The following example forces the PlanAhead tool to reimport all project files regardless of whether they are out of date, or the local files are newer than the referenced source file:

```
reimport_files -force
```

Note: No warnings will be issued for newer local files that will be overwritten.

The following example reimports the specified files to the project:

```
reimport_files C:/Data/FPGA_Design/source1.v C:/Data/FPGA_Design/source2.vhdl
```

Only the two specified files will be reimported, and only if the original source file is newer than the local project file.

See Also

- add_files
- import_files

remove_cells_from_pblock

Remove cells from a Pblock

Syntax

```
remove_cells_from_pblock [-quiet] pblock cells ...
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
pblock	no		Pblock to remove cells from
cells	no		Cells to remove

Categories

Floorplan

Description

This command removes the specified logic instances from a Pblock. Cells are added to a Pblock with the add_cells_to_pblock command.

Cells that have been placed will not be unplaced as they are removed from a Pblock. ANy current LOC assignments are left intact.

Arguments

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

pblock - Specify the name of the Pblock to remove the specified instances from.

cells - Specify one or more cell objects to remove from the specified Pblock.

Example

The following example removes the specified cells from the pb_cpuEngine Pblock:

```
remove_cells_from_pblock pb_cpuEngine [get_cells cpuEngine/cpu_dwb_dat_o/*]
```

See Also

add_cells_to_pblock

remove_clock

Remove clock object(s)

Syntax

```
remove_clock [-all] [-quiet] [clocks]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-all	yes		Remove all clocks
-quiet	yes		Ignore command errors
clocks	yes		list of clocks

Categories

SDC, XDC

remove_clock_latency

Removes actual or predicted clock latency

Syntax

```
remove_clock_latency [-clock args] [-source] [-quiet] objects
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-clock	yes		List of relative clocks
-source	yes		Specify clock rise and fall source latency
-quiet	yes		Ignore command errors
objects	no		List of clocks, ports or pins

Categories

SDC, XDC

remove_data_check

Remove data to data checks

Syntax

```
remove_data_check [-from args] [-to args] [-rise_from args]
[-fall_from args] [-rise_to args] [-fall_to args] [-setup]
[-hold] [-clock args] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-from	yes		From pin/port of data to data check
-to	yes		To pin/port of the data to data check
-rise_from	yes		Rise from pin/port of data to data check
-fall_from	yes		Fall from pin/port of data to data check
-rise_to	yes		Rise to pin/port of data to data check
-fall_to	yes		Fall to pin/port of data to data check
-setup	yes		Specify data check setup time
-hold	yes		Specify data check hold time
-clock	yes		Specify the clock domain at related pin/port of the checks
-quiet	yes		Ignore command errors

Categories

SDC, XDC

Description

This command specifies data-to-data checks are removed between the **-from** object and the **-to** object for the specified options. Data-to-data checks are defined using the **set_data_check** command.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-from *value* - (Optional) Remove the data-to-data check for data traveling from the specified port.

-to *value* - (Optional) Remove the data-to-data check for data traveling to the specified port.

-rise_from *value* - (Optional) Remove the data-to-data check for the rising edge traveling from the specified port.

-rise_to *value* - (Optional) Remove the data-to-data check for the rising edge traveling to the specified port.

-fall_from *value* - (Optional) Remove the data-to-data check for the falling edge traveling from the specified port.

-fall_to *value* - (Optional) Remove the data-to-data check for the falling edge traveling to the specified port.

-setup - (Optional) Remove setup time check.

-hold - (Optional) Remove hold time check.

-clock *value* - (Optional) Specify the clock domain at related pin/port of the checks.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example removes a data-to-data check from and1/B to and1/A with respect to the rising edge of signals at and1/B. Both setup and hold checks are removed.

```
remove_data_check -rise_from and1/B -to and1/A
```

The following example removes setup data-to-data check between and1/B to and1/A with respect to the rising edge of signals at and1/B, coming from start points triggered with clock CK1, falling edge of signals at and1/A.

```
remove_data_check -rise_from and1/B -fall_to and1/A -setup -clock [get_clock CK1]
```

See Also

report_timing

remove_disable_timing

Enable timing arcs

Syntax

```
remove_disable_timing [-from arg] [-to arg] [-quiet] objects
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-from	yes		From pin on cell
-to	yes		To pin on cell
-quiet	yes		Ignore command errors
objects	no		List of cells or pins, ports, lib-cells, lib-pins, libcell/cell timing-arcs

Categories

XDC

Description

This command is used to enable timing arcs. This command removes a disabled timing constraint by specifying the arguments. The command contains "from" and "to" arguments. The from argument determines the start point. The to argument is the end point.

Note: This command operates silently and does not return direct feedback of its operation

Arguments

-from *value* - (Optional) From Pin on Cell

-to *value* - (Optional) To Pin on Cell

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

objects - (Optional) List of cells or pins, ports, lib-cells, lib-pins, libcell/cell timing-arcs

Example

The above command removes a disabled timing constraint in between the start point EN_A to D_IN.

```
remove_disable_timing -from EN_A -to D_IN
```

See Also

report_timing

remove_files

Remove files or directories from a fileset

Syntax

```
remove_files [-fileset arg] [-quiet] [files ...]
```

Returns

list of files that were removed

Usage

Name	Optional	Default	Description
-fileset	yes		Fileset name
-quiet	yes		Ignore command errors
files	yes		Name of the files and/or directories to remove

Categories

Project, Simulation

Description

Removes files from the specified fileset. Files must be specified with the full path to the file.

Arguments

-fileset *arg* - Specifies the name of the fileset to remove files from. If no fileset is specified, the PlanAhead tool will attempt to remove the specified file from the `source_1` fileset.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

files - Specify one or more files to be removed from the project. The file must be specified with the full path to the file.

Note: If the path is not specified as part of the file name, the PlanAhead tool will search for the specified file in the current working directory, or the directory from which the PlanAhead tool was launched.

Examples

The following example removes the file named `C:/Design/top.ucf` from the constraint set `constrs_1`:

```
remove_files -fileset constrs_1 C:/Design/top.ucf
```

Multiple files can be specified as follows:

```
remove_files -fileset sim_1 top_tb1.vhdl top_tb2.vhdl
```

See Also

add_files

remove_propagated_clock

Remove a propagated clock

Syntax

```
remove_propagated_clock [-quiet] objects
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
objects	no		List of clocks, ports, or pins

Categories

SDC, XDC

reorder_files

Change the order of source files in the active fileset

Syntax

```
reorder_files [-fileset arg] [-before arg] [-after arg]
[-front] [-back] [-auto] [-disable_unused] [-quiet] files ...
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-fileset	yes		Fileset to reorder
-before	yes		Move the listed files before this file
-after	yes		Move the listed files after this file
-front	yes		Move the listed files to the front (default)
-back	yes		Move the listed files to the back
-auto	yes		Automatically re-orders the given fileset
-disable_unused	yes		Disables all files not associated with the TOP design unit
-quiet	yes		Ignore command errors
files	no		Files to move

Categories

Project

Description

Reorders source files in the specified fileset. Takes the files indicated and places them at the front of, the back of, or before or after other files within the fileset. This command also has an auto reorder feature that allows the PlanAhead tool to reorder the files based on the requirements of the current top module in the design.

Arguments

-fileset *arg* - Specifies the fileset to reorder files in. The default is the sources_1 source fileset.

-before *arg* - Place the specified files before this file in the fileset. The file must be specified with the full path name in the fileset, or the PlanAhead tool will not find the file.

-after *arg* - Place the specified files after this file in the fileset. The file must be specified with the full path name in the fileset, or the PlanAhead tool will not find the file.

-front - Place the specified files at the front of the list of files in the fileset.

-back - Place the specified files at the back of the list of files in the fileset.

-auto - Enables the PlanAhead tool to reorder files automatically based on the hierarchy requirements of the current top-module in the project. This argument is often used after changing the top module with the "**set_property top**" command.

-disable_unused - This argument allows the PlanAhead tool to disable any files not currently used by the hierarchy based on the top-module. This argument is often used after changing the top module with the "**set_property top**" command.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

files - Specifies one or more files to relocate in the fileset. These files must be specified by their full path name in the fileset, and are reordered in the order they are specified here.

Examples

The following example takes the specified files and moves them to the front of the source fileset:

```
reorder_files -front {C:/Data/FPGA/file1.vhdl C:/Data/FPGA/file2.vhdl}
```

Note that the default source fileset is used in the preceding example since the **-fileset** argument is not specified.

The following example sets a new top_module in the design, and then has the PlanAhead tool automatically reorder and disable unused files based on the hierarchy of the new top-module:

```
set_property top block1 [current_fileset]
reorder_files -auto -disable_unused
```

See Also

- add_files
- create_fileset
- current_fileset
- remove_files

report_clock_interaction

Report on inter clock timing paths and unclocked registers

Syntax

```
report_clock_interaction [-delay_type arg] [-setup] [-hold]
[-significant_digits arg] [-file arg] [-append] [-name arg]
[-return_string] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-delay_type	yes	max	Type of path delay: Values: max, min, min_max
-setup	yes		Consider max delay timing paths (equivalent to -delay_type max)
-hold	yes		Consider min delay timing paths (equivalent to -delay_type min)
-significant_digits	yes	2	Number of digits to display: Range: 0 to 13
-file	yes		Filename to output results to. (send output to console if -file is not used)
-append	yes		Append the results to file, don't overwrite the results file
-name	yes		Output the results to GUI panel with this name
-return_string	yes		Return report as string
-quiet	yes		Ignore command errors

Categories

Report

Description

Report clock interactions and signals that cross clock domains to identify potential problems such a metastability or data loss or incoherency some visibility into the paths that cross clock domains is beneficial.

This command requires an open design.

Arguments

-delay_type *arg* - Specifies the type of delay to analyze when running the clock interaction report. The valid values are min, max, and min_max. The default setting for **-delay_type** is max.

-setup - Check for setup violations. This is the same as specifying **-delay_type max**.

-hold - Check for hold violations. This is the same as specifying **-delay_type min**.

Note: **-setup** and **-hold** can be specified together, which is the same as specifying **-delay_type min_max**

-significant_digits *arg* - Specifies the number of significant digits in the output results. The valid range is 0 to 13. The default setting is 2 significant digits.

-file *arg* - This option tells the PlanAhead tool to output the Clock Interaction Report to the specified filename. If the path is not specified as part of the file name the file will be created in the current working directory, or the directory from which the PlanAhead tool was launched.

-append - Append the output of the **report_clock_interaction** command to the specified file rather than overwriting it.

Note: The **-append** option can only be used with the **-file** option

-name *arg* - Specify the name of the Clock Interaction Report view to display in the PlanAhead tool GUI mode. If the name has already been used in an open Report view, that view will be closed and a new report opened.

-return_string - Directs the output to a Tcl string. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example sets the model for interconnect delay, selects a device speed grade, and then runs **report_clock_interaction**:

```
set_delay_model -interconnect none
set_speed_grade -3
report_clock_interaction -delay_type min_max -significant_digits 3 -name "results_1"
```

The following example returns the clock interactions, writing the report to the GUI, to the specified file, and returns a string which is assigned to the specified variable:

```
set clk_int [report_clock_interaction -file clk_int.txt -name clk_int1 \ -return_string]
```

See Also

- create_clock
- create_generated_clock
- report_clocks
- set_delay_model
- set_speed_grade

report_clock_utilization

Report information about clock nets in design

Syntax

```
 report_clock_utilization [-file arg] [-write_xdc arg]
[-return_string] [-quiet]
```

Returns

Report

Usage

Name	Optional	Default	Description
-file	yes		Filename to output results to. (send output to console if -file is not used)
-write_xdc	yes		file to output clock constraint. If not specified the clock constraint will be appended to clock report.
-return_string	yes		return report as string
-quiet	yes		Ignore command errors

Categories

Report

Description

Returns information related to clock nets in the design and clock resource utilization on the target device.

By default the **report_clock_utilization** command puts the output to the standard output. However, you can redirect the output of the command to either a file or to a string for further processing.

Arguments

-file *arg* - This option tells the PlanAhead tool to output the clock utilization report to the specified filename. If the path is not specified as part of the file name the file will be created in the current working directory, or the directory from which the PlanAhead tool was launched.

-write_xdc *arg* - This option tells the PlanAhead tool to output XDC location constraints for the various clock resources to the specified filename. If the path is not specified as part of the file name the file will be created in the current working directory, or the directory from which the PlanAhead tool was launched.

Note: The XDC constraints in the clock utilization report begin with the "Location of..." statement. If the **write_xdc** option is specified, these lines will be output to the specified file rather than to the standard output

-return_string - Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example returns information about the clock nets in the design and the clock resources utilized on the target device, and writes it to the specified file:

```
report_clock_utilization -file C:/Data/FPGA_Design/clock_util.txt
```

The following example reports the clock nets and clock resource utilization to the standard output, but writes the XDC location constraints to the specified file:

```
report_clock_utilization -write_xdc clock_util_xdc.txt
```

Because the path is not specified as part of the XDC file name, the file will be created in the current working directory, or the directory from which the PlanAhead tool was launched.

See Also

- create_clock
- create_generated_clock

report_clocknetworks

Report clock networks

Syntax

```
report_clocknetworks [-file arg] [-append] [-name arg]
[-return_string] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-file	yes		Filename to output results to. (send output to console if -file is not used)
-append	yes		Append the results to file, don't overwrite the results file
-name	yes		Output the results to GUI pannel with this name
-return_string	yes		return report as string
-quiet	yes		Ignore command errors

Categories

Report

report_clocks

Report clocks

Syntax

```
report_clocks [-file arg] [-append] [-return_string] [-quiet] [clocks]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-file	yes		Filename to output results to. (send output to console if -file is not used)
-append	yes		Append the results to file, don't overwrite the results file
-return_string	yes		return report as string
-quiet	yes		Ignore command errors
clocks	yes	*	List of clocks

Categories

Report

Description

Returns a table showing all the clocks in a design, including propagated clocks, generated and auto-generated clocks, virtual clocks, and inverted clocks. More detailed information about each clock net can be obtained with the **report_clock_utilization** command.

By default the **report_clocks** command puts the output to the standard output. However, you can redirect the output of the command to either a file or to a string for further processing.

Arguments

-file *arg* - This option tells the PlanAhead tool to output the clock report to the specified filename. If the path is not specified as part of the file name the file will be created in the current working directory, or the directory from which the PlanAhead tool was launched.

-append - Append the output of the report_clocks command to the specified file rather than overwriting it.

Note: The **-append** option can only be used with the **-file** option

-return_string - Directs the output to a Tcl string rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

clocks - Match clocks against the specified patterns. The default pattern is the wildcard '*' which returns all clocks in the design. More than one pattern can be specified to find multiple clocks based on different search criteria.

Examples

The following example returns the name, period, waveform, and sources of the clocks in the current design:

```
report_clocks -file C:/Data/FPGA_Design/clock_out.txt
```

The following example reports the clocks in the design with "Clock" in the name:

```
report_clocks *Clock*
```

See Also

- create_clock
- create_generated_clock
- report_clock_utilization

report_config_timing

Report settings affecting timing analysis

Syntax

```
report_config_timing [-file arg] [-append] [-name arg]
[-return_string] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-file	yes		Output the results to file
-append	yes		Append the results to file, don't overwrite the results file
-name	yes		Output the results to GUI panel with this name
-return_string	yes		return report as string
-quiet	yes		Ignore command errors

Categories

Report

report_constraints

Displays constraint-related information about a design.

Syntax

```
report_constraints [-file arg] [-append] [-return_string]
[-all_violators] [-verbose] [-path_type arg] [-max_delay]
[-min_delay] [-recovery] [-removal] [-clock_gating_setup]
[-clock_gating_hold] [-min_pulse_width] [-min_period]
[-significant_digits arg] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-file	yes		Filename to output results to. (send output to console if -file is not used)
-append	yes		Append the results to file, don't overwrite the results file
-return_string	yes		return report as string
-all_violators	yes		Show all constraint violators
-verbose	yes		Output verbose information
-path_type	yes	slack_only	Format for path report: Allowed values are 'end' and 'slack_only'
-max_delay	yes		Only max_delay & setup
-min_delay	yes		Only min_delay & hold
-recovery	yes		Only async recovery
-removal	yes		Only async removal
-clock_gating_setup	yes		Only clock gating setup
-clock_gating_hold	yes		Only clock gating hold
-min_pulse_width	yes		Only min_pulse_width
-min_period	yes		Only min_period
-significant_digits	yes	2	Number of digits to display: Range: 0 to 13
-quiet	yes		Ignore command errors

Categories

Report

report_control_sets

Report the unique control sets in design

Syntax

```
 report_control_sets [-file arg] [-verbose] [-sort_by args]
[-cells args] [-return_string] [-quiet]
```

Returns

Report

Usage

Name	Optional	Default	Description
-file	yes		Filename to output results to. (send output to console if -file is not used)
-verbose	yes		Output verbose information
-sort_by	yes		Sort criterion. Used with -verbose switch
-cells	yes		Cells/bel_instances for which to report control sets
-return_string	yes		return report as string
-quiet	yes		Ignore command errors

Categories

Report

report_debug_core

Report details on ChipScope debug cores

Syntax

```
report_debug_core [-file arg] [-return_string] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-file	yes		Filename to output results to. (send output to console if -file is not used)
-return_string	yes		Return report as a string
-quiet	yes		Ignore command errors

Categories

Report, ChipScope

Description

Writes a report of the various ChipScope debug cores in the current project, and the parameters of those cores. Debug cores can be added to a project using the create_debug_core command or the read_chipscope_cdc command.

The report is written to the Tcl console or STD output as a default. However, the results can also be written to a file or returned as a string if desired.

Arguments

-file *arg* - Specify a file name to save the debug core report to.

Note: If the path is not specified as part of the file name, the PlanAhead tool will write the named file into the current working directory, or the directory from which the PlanAhead tool was launched.

-return_string - Return report as a string. This argument can not be used with the -file argument.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example writes the debug core report to the specified file name at the specified location:

```
report_debug_core -file C:/Data/FPGA_Design/project_1_cores.txt
```

See Also

- create_debug_core
- read_chipscope_cdc

report_disable_timing

Report disabled timing arcs

Syntax

```
report_disable_timing [-file arg] [-append] [-return_string]
[-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-file	yes		Filename to output results to. (send output to console if -file is not used)
-append	yes		Append the results to file, don't overwrite the results file
-return_string	yes		return report as string
-quiet	yes		Ignore command errors

Categories

Report

report_drc

Run DRC

Syntax

```
report_drc [-name arg] [-list_rules] [-rules args] [-file arg]
[-return_string] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-name	yes		Output the results to GUI panel with this name
-list_rules	yes		List available rules to pass to -rules option
-rules	yes		DRC rules (see -list_rules for available rules)
-file	yes		DRC Report file
-return_string	yes		return report as string
-quiet	yes		Ignore command errors

Categories

Report

report_io

Display information about all the IO sites on the device

Syntax

```
report_io [-file arg] [-return_string] [-quiet]
```

Returns

Report

Usage

Name	Optional	Default	Description
-file	yes		Filename to output results to. Send output to console if -file is not used.
-return_string	yes		return report as string
-quiet	yes		Ignore command errors

Categories

Report

report_min_pulse_width

Report min pulse width check

Syntax

```
report_min_pulse_width [-file arg] [-append] [-return_string]
[-path_type arg] [-significant_digits arg] [-input_pins]
[-verbose] [-quiet] [objects]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-file	yes		Filename to output results to. (send output to console if -file is not used)
-append	yes		Append the results to file, don't overwrite the results file
-return_string	yes		return report as string
-path_type	yes	summary	Format for path report: Allowed values are 'summary', 'short', 'full_clock' and 'full_clock_expanded '
-significant_digits	yes	2	Number of digits to display: Range: 0 to 13
-input_pins	yes		Show input pins in path
-verbose	yes		Output verbose information
-quiet	yes		Ignore command errors
objects	yes		List of objects to check min pulse width with

Categories

Report

report_param

Get information about all parameters

Syntax

```
report_param [-quiet] [pattern]
```

Returns

param report

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
pattern	yes	*	Display params matching pattern

Categories

PropertyAndParameter, Report

Description

Returns a list of all user-definable parameters in the PlanAhead tool, the current value, and a description of what the parameter configures or controls.

Arguments

pattern - Match parameters against the specified pattern. The default pattern is the wildcard '*' which returns all user-definable parameters.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example returns the name, value, and description of all user-definable parameters:

```
report_param
```

The following example returns the name, value, and description of user-definable parameters that match the specified search pattern:

```
report_param *coll*
```

See Also

- get_param
- list_param
- reset_param
- set_param

report_power

Run power estimation and display report

Syntax

```
report_power [-verbose] [-hierarchy] [-levels arg]
[-file arg] [-results arg] [-format arg] [-xpe arg] [-console]
[-return_string] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-verbose	yes		Verbose power estimation reporting
-hierarchy	yes		Report power consumption hierarchically
-levels	yes	1	Number of levels of hierarchy to be reported: Value = 1
-file	yes		Filename to output results to. (send output to console if -file is not used)
-results	yes		Name to store results
-format	yes	table	Format for the power estimation report: table, xml
-xpe	yes		Output the results to XML file for importing into XPE
-console	yes		Display results on console
-return_string	yes		return report as string
-quiet	yes		Ignore command errors

Categories

Report, Power

report_property

Report properties of object

Syntax

```
report_property [-all] [-return_string] [-quiet] object
```

Returns

property report

Usage

Name	Optional	Default	Description
-all	yes		Report all properties of object even if not set
-return_string	yes		Set the result of running report_property in the Tcl interpreter's result variable
-quiet	yes		Ignore command errors
object	no		Object to query for properties

Categories

Object, PropertyAndParameter, Report

Description

This command takes an object as input and returns information regarding all properties on the object. The information returned is the property name, property type, and property value.

Note: list_property also returns a list of properties on an object, but does not include the property type or value.

Arguments

-all - Specifies that all properties of an object should be returned, even if the property value is not defined.

object - Specifies one object to report properties.

Note: If you specify multiple objects the PlanAhead tool will return an error.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example returns all properties of the specified object:

```
report_property -all [get_cells cpuEngine]
```

See Also

- create_property
- get_cells
- get_property
- list_property
- list_property_value
- reset_property
- set_property

report_resources

Run resource estimation and display report

Syntax

```
report_resources [-verbose] [-hierarchical] [-levels arg]
[-file arg] [-return_string] [-format arg] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-verbose	yes		Output verbose information
-hierarchical	yes		Report estimates hierarchically
-levels	yes	1	Number of levels of hierarchy to be reported: Value = 1
-file	yes		Filename to output results (send output to console if -file is not used)
-return_string	yes		Return report as string
-format	yes	table	Format for the resource estimation report: table, xml
-quiet	yes		Ignore command errors

Categories

Report

report_route_status

Report on status of routing. By default it reports total number of physical nets, number of routed, unrouted, and partially routed nets. When a route object is passed in it reports the connectivity of that route, showing the names of each node and it's connectivity.

Syntax

```
report_route_status [-of_objects args] [-return_string]
[-file arg] [-append] [-verbose] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-of_objects	yes		Report detailed routing for these routes
-return_string	yes		Set the result of running the report in the Tcl interpreter's result variable
-file	yes		Filename to output results to. (send output to console if -file is not used)
-append	yes		Append the results to file, don't overwrite the results file
-verbose	yes		show verbose routing information for each net in the design
-quiet	yes		Ignore command errors

Categories

Report

report_ssn

Run SSN analysis on the current package and pinout

Syntax

```
report_ssn [-name arg] [-return_string] [-file arg] [-quiet]
```

Returns

ssn report

Usage

Name	Optional	Default	Description
-name	yes		Output the results to GUI panel with this name
-return_string	yes		Return report as string
-file	yes		Filename to output results to. (send output to console if -file is not used)
-quiet	yes		Ignore command errors

Categories

Report

report_sso

Run WASSO analysis on the current package and pinout

Syntax

```
report_sso -name arg [-return_string] [-file arg]
[-board_thickness arg] [-via_diameter arg]
[-pad_to_via_breakout_length arg] [-breakout_width arg]
[-other_pcb_inductance arg] [-socket_inductance arg]
[-ground_bounce arg] [-output_cap arg] [-quiet]
```

Returns

sso report

Usage

Name	Optional	Default	Description
-name	no		Output the results to GUI panel with this name
-return_string	yes		Return report as string
-file	yes		Filename to output results to. (send output to console if -file is not used)
-board_thickness	yes		Thickness of the PCB in mils
-via_diameter	yes		Finished via diameter in mils
-pad_to_via_breakout_length	yes		Pad to via breakout length in mils
-breakout_width	yes		Breakout width in mils
-other_pcb_inductance	yes		Other PCB parasitic inductance in nanohenrys
-socket_inductance	yes		Socket inductance in nanohenrys
-ground_bounce	yes		Maximum ground bounce in millivolts
-output_cap	yes		Capacitance per output driver in picofarads
-quiet	yes		Ignore command errors

Categories

Report

report_stats

Report Statistics

Syntax

```
report_stats [-file arg] [-cell arg] [-pblock arg]
[-clock_region arg] [-format arg] [-level arg] [-all]
[-tables args] [-quiet]
```

Returns

Report

Usage

Name	Optional	Default	Description
-file	yes		Output filename; writes console if not spacified
-cell	yes		Write statistics for the specified cell
-pblock	yes		Write statistics for the specified Pblock
-clock_region	yes		Write statistics for the specified clock region
-format	yes	TABLE	Report format Values: TABLE, CSV, XML
-level	yes	1	Report level (used with '-cell' or '-pblock')
-all	yes		Report all levels (used with '-cell' or '-pblock')
-tables	yes		List of table types Values: rtlMacro, rtlPrimitive, rtlHierarchy, rtlMemory, rtlPower, rtlPower2, primitive, netBoundary, carryChain, physicalResource, io, RPM, clock, PRModule, PRModule, pblockOverlap, ioBank
-quiet	yes		Ignore command errors

Categories

Report

report_timing

Report timing paths

Syntax

```
report_timing [-from args] [-rise_from args] [-fall_from args]
[-to args] [-rise_to args] [-fall_to args] [-through args]
[-rise_through args] [-fall_through args] [-delay_type arg]
[-setup] [-hold] [-max_paths arg] [-nworst arg]
[-path_type arg] [-input_pins] [-nets] [-slack_lesser_than arg]
[-slack_greater_than arg] [-group args] [-sort_by arg]
[-significant_digits arg] [-no_report_unconstrained] [-file arg]
[-append] [-match_style arg] [-name arg] [-return_string]
[-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-from	yes		From pins, ports, cells or clocks
-rise_from	yes		Rising from pins, ports, cells or clocks
-fall_from	yes		Falling from pins, ports, cells or clocks
-to	yes		To pins, ports, cells or clocks
-rise_to	yes		Rising to pins, ports, cells or clocks
-fall_to	yes		Falling to pins, ports, cells or clocks
-through	yes		Through pins, ports, cells or nets
-rise_through	yes		Rising through pins, ports, cells or nets
-fall_through	yes		Falling through pins, ports, cells or nets
-delay_type	yes	max	Type of path delay: Values: max, min, min_max, max_rise, max_fall, min_rise, min_fall
-setup	yes		Report max delay timing paths (equivalent to -delay_type max)
-hold	yes		Report min delay timing paths (equivalent to -delay_type min)

Name	Optional	Default	Description
-max_paths	yes	1	Maximum number of paths to output when sorted by slack, or per path group when sorted by group: Value =1
-nworst	yes	1	List N worst paths to endpoint: Value =1
-path_type	yes	full_clock_expanded	Format for path report: Values: end summary short full full_clock full_clock_expanded
-input_pins	yes		Show input pins in path
-nets	yes		List net names
-slack_lesser_than	yes	1e+30	Display paths with slack less than this
-slack_greater_than	yes	-1e+30	Display paths with slack greater than this
-group	yes		Limit report to paths in this group(s)
-sort_by	yes	slack	Sorting order of paths: Values: group, slack
-significant_digits	yes	3	Number of digits to display: Range: 0 to 13
-no_report _unconstrained	yes		Do not report unconstrained paths
-file	yes		Filename to output results to. (send output to console if -file is not used)
-append	yes		Append the results to file, don't overwrite the results file
-match_style	yes	ucf	Style of pattern matching, valid values are ucf, sdc
-name	yes		Output the results to GUI panel with this name
-return_string	yes		return report as string
-quiet	yes		Ignore command errors

Categories

Report

report_transformed_primitives

Report details of Unisim primitive transformations.

Syntax

```
report_transformed_primitives [-file arg] [-return_string]
[-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-file	yes		Output file
-return_string	yes		return report as string
-quiet	yes		Ignore command errors

Categories

Report

report_utilization

Compute utilization of device and display report

Syntax

```
report_utilization [-file arg] [-append] [-pblocks args]
[-cells args] [-return_string] [-quiet]
```

Returns

Report

Usage

Name	Optional	Default	Description
-file	yes		Filename to output results to. (send output to console if -file is not used)
-append	yes		Append the results to file, don't overwrite the results file
-pblocks	yes		Report utilization of given list of pblocks
-cells	yes		Report utilization of given list of cells
-return_string	yes		Return report as string
-quiet	yes		Ignore command errors

Categories

Report

reset_default_switching_activity

Reset switching activity on default types

Syntax

```
 reset_default_switching_activity [-static_probability]
[-toggle_rate] -type args [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-static_probability	yes		reset static probability
-toggle_rate	yes		reset toggle rate
-type	no		Resets the default seed values to tool defaults on specified types for vector-less propagation engine. List of valid default type values: input, input_set, input_reset, input_enable, register, dsp, bram_read_enable, bram_write_enable, output_enable, clock, all
-quiet	yes		Ignore command errors

Categories

XDC

Description

This command is used to reset the attributes of the default switching activity on nets, ports, pins, and cells in the design.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-static_probability - (Optional) Reset the static probability of the specified type.

-toggle_rate - (Optional) Reset the toggle rate of the specified type.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

type - (Required) Specifies the type to reset. List of valid values: input, input_set, input_reset, input_enable, register, dsp, bram_read_enable, bram_write_enable, output_enable, clock, all.

Example

The following example resets the toggle rate and static probability value on all design output ports:

```
reset_default_switching_activity -toggle_rate -static_probability all
```

See Also

- report_power
- reset_switching_activity
- set_default_switching_activity
- set_switching_activity

reset_drc

Remove DRC

Syntax

```
reset_drc [-quiet] [name]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
name	yes		DRC result name

Categories

Report

reset_ip

Reset a configurable IP

Syntax

```
reset_ip [-srcset arg] -files args [-quiet]
```

Returns

list of files that were reset

Usage

Name	Optional	Default	Description
-srcset	yes		Source set name
-files	no		IP source files to be reset
-quiet	yes		Ignore command errors

Categories

COREGenerator

reset_msg_limit

Reset message limit, return message limit

Syntax

```
reset_msg_limit [-severity arg] [-id arg] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-severity	yes	ALL	Message severity to be reset (not valid with -id,) e.g. "ERROR" or "CRITICAL WARNING"
-id	yes		Unique message Id to be reset (not valid with -severity,) e.g Common-99
-quiet	yes		Ignore command errors

Categories

Report

Description

This command restores the default message limit for messages returned by the PlanAhead tool. The command can be used to restore the default limit for a specific message ID, for a specific message severity, or for all messages returned by the PlanAhead tool. The current default limit for all messages returned is 4,294,967,295.

By default this command returns the message limit for all messages. You can also get the limit of a specific severity of message, or for a specific message ID. For instance the following are two messages returned by the PlanAhead tool under different circumstances:

INFO: [common-99] This is an example INFO message

CRITICAL WARNING: [Netlist-1129] This message is a CRITICAL WARNING and requires user attention

Note: You can change the severity of a specific message ID with the **set_msg_severity** command.

Arguments

-id *value* - is found in the PlanAhead tool in the Messages view or other reports when the message is reported. Use the specific message ID for the message of interest for use in this command. The message IDs above are common-99 and Netlist-1129.

-severity *value* - Specifies the severity of the message. There are five message severities:

- **ERROR** - An ERROR condition implies an issue has been encountered which will render design results unusable and cannot be resolved without user intervention.
- **{CRITICAL WARNING}** - A CRITICAL WARNING message indicates that certain input/constraints will either not be applied or are outside the best practices for a FPGA family. User action is strongly recommended. Note: Since this is a two word value, it must be enclosed in {}.
- **WARNING** - A WARNING message indicates that design results may be sub-optimal because constraints or specifications may not be applied as intended. User action may be taken or may be reserved.
- **INFO** - An INFO message is the same as a STATUS message, but includes a severity and message ID tag. An INFO message includes a message ID to allow further investigation through answer records if needed.
- **STATUS** - A STATUS message communicates general status of the process and feedback to the user regarding design processing. A STATUS message does not include a message ID.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example resets the default limit for all messages:

```
reset_msg_limit
```

Note: The default limit is 4,294,967,295.

The following example resets the message limit of the specified message ID:

```
reset_msg_limit -id Netlist-1129
```

See Also

- get_msg_limit
- set_msg_limit
- set_msg_severity

reset_msg_severity

Reset Message Severity by ID

Syntax

```
reset_msg_severity [-quiet] id
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
id	no		Unique message id to be set, e.g. "Common-99"

Categories

Project

Description

Restores the specified message ID to the default setting provided by the PlanAhead tool. Use this command after **set_msg_severity** to restore a specific message ID to its original severity level.

Arguments

id - is found in the PlanAhead tool in the Messages view or other reports when the message is reported. Use the message ID of a specific message to restore its severity. For instance, following are two messages returned by the PlanAhead tool under different conditions:

INFO: [common-99] This is an example INFO message

CRITICAL WARNING: [Netlist-1129] This message is a CRITICAL WARNING and requires user attention

The message IDs for these two messages are common-99 and Netlist-1129.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example restores the message ID **common-99** to its original severity:

```
reset_msg_severity common-99
```

The following example resets the severity of message ID **Netlist-1129**:

```
reset_msg_severity Netlist-1129
```

See Also

set_msg_severity

reset_operating_conditions

Reset operating conditions to tool default for power estimation

Syntax

```
reset_operating_conditions [-voltage args] [-grade] [-process]
[-junction_temp] [-ambient_temp] [-thetaja] [-thetasa] [-airflow]
[-heatsink] [-thetajb] [-board] [-board_temp] [-board_layers]
[-all] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-voltage	yes		Resets voltage value. Supported voltage supplies vary by family.
-grade	yes		Resets temperature grade
-process	yes		Resets process
-junction_temp	yes		Resets Junction Temperature
-ambient_temp	yes		Resets Ambient Temperature
-thetaja	yes		Resets ThetaJA
-thetasa	yes		Resets ThetaSA
-airflow	yes		Resets Airflow
-heatsink	yes		Resets dimensions of heatsink
-thetajb	yes		Resets ThetaJB
-board	yes		Resets Board type
-board_temp	yes		Resets Board Temperature
-board_layers	yes		Resets Board layers
-all	yes		Resets all operating conditions listed in this command line
-quiet	yes		Ignore command errors

Categories

SDC, XDC

Description

The environmental operating conditions of the device are used for power analysis when running the **report_power** command, but are not used during timing analysis. This command resets the specified operating conditions to their default values.

The operating conditions can be set using the **set_operating_conditions** command. The current values can be determined using the **report_operating_conditions** command.

Note: This command operates silently and does not return direct feedback of its operation

Arguments

-voltage *args* - (Optional) Reset the voltage pairs to the specified value.

-grade - (Optional) Reset the grade of the selected device. The default value is "commercial".

-process - (Optional) Reset the manufacturing process for the target device. The default process is "typical".

-junction_temp - (Optional) Reset the junction temp for the target device. The default value is "auto".

-ambient_temp - (Optional) Reset the ambient temp of the design. The default setting is "default".

-thetaja - (Optional) Reset the Theta-JA thermal resistance. The default setting is "auto".

-thetasa - (Optional) Reset the Theta-SA thermal resistance. The default setting is "auto".

-airflow - (Optional) Reset the Linear Feet Per Minute (LFM) airflow. The default setting varies by device family.

-heatsink - (Optional) Reset the heatsink profile. The default setting is "medium".

-thetajb - (Optional) Reset the Theta-JB thermal resistance. The default setting is "auto".

-board - (Optional) Reset the board size to be used for modeling. The default value is "medium".

-board_temp arg - (Optional) Reset the board temperature to the default setting.

-board_layers - (Optional) Reset the number of board layers to be used for modeling to the default setting of "8to11".

-all - (Optional) Reset all operating conditions to their default value. Use this to avoid having to reset each condition separately.

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example resets the junction, ambient, and board temperature for the design to their default settings:

```
reset_operating_conditions -junction_temp -ambient_temp -board_temp
```

See Also

set_operating_conditions

reset_param

Reset a parameter

Syntax

```
reset_param [-quiet] name
```

Returns

original value

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
name	no		Parameter name

Categories

PropertyAndParameter

Description

Use this command to reset any user-definable configuration parameter within the PlanAhead tool that has been changed with the **set_param** command. The parameter specified will be restored to its default value.

You can use the **report_param** command to see what parameters are currently defined.

Arguments

name - Specifies the name of a parameter to reset the value to its default setting. This command only accepts one parameter at a time.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example restores the specified parameter to its default value:

```
reset_param tcl.statsThreshold
```

See Also

- get_param
- list_param
- report_param
- set_param

reset_path

Resets specified paths to single cycle behavior

Syntax

```
 reset_path [-setup] [-hold] [-rise] [-fall] [-from args]
[-rise_from args] [-fall_from args] [-to args] [-rise_to args]
[-fall_to args] [-through args] [-rise_through args]
[-fall_through args] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-setup	yes		Reset setup timing analysis for paths
-hold	yes		Reset hold timing analysis for paths
-rise	yes		Reset only rising delays for the defined paths
-fall	yes		Reset only falling delays for the defined paths
-from	yes		List of path startpoints or clocks
-rise_from	yes		Apply to paths rising from the list of startpoints or clocks
-fall_from	yes		Apply to paths falling from the list of startpoints or clocks
-to	yes		List of path endpoints or clocks
-rise_to	yes		Apply to paths with rise transition at the list of endpoints or clocks
-fall_to	yes		Apply to paths with fall transition at the list of endpoints or clocks
-through	yes		List of through pins, cells or nets
-rise_through	yes		Apply to paths rising through pins, cells or nets
-fall_through	yes		Apply to paths falling through pins, cells or nets
-quiet	yes		Ignore command errors

Categories

Report

reset_property

Reset property on object(s)

Syntax

```
reset_property [-quiet] property_name objects ...
```

Returns

The value that was set if success, "" if failure

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
property_name	no		Name of property to reset
objects	no		Objects to set properties

Categories

XDC, Object, PartialReconfiguration, PropertyAndParameter, Partition

Description

Resets the specified property to its default value on the specified object or objects. If no default is defined for the property, the property is unassigned on the specified object.

Arguments

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

property_name - Specifies the name of the the property to be reset.

objects - Specifies one or more objects to assign the property to.

Example

The following example resets the ALL_PROPS property on all cells:

```
reset_property ALL_PROPS [get_cells]
```

See Also

- create_property
- get_cells
- get_property
- list_property
- list_property_value
- report_property
- set_property

reset_run

Reset an existing run

Syntax

```
reset_run [-from_step arg] [-noclean_dir] [-quiet] run
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-from_step	yes		First Step to reset
-noclean_dir	yes		Do not remove all output files and directories from disk
-quiet	yes		Ignore command errors
run	no		Run to modify

Categories

Project

Description

Reset the specified run to an unimplemented or unsynthesized state. Use this command to reset the run to prepare it to be run again.

Arguments

-prev_step - Reset an implementation run from the last step completed. This can be used to reset an implementation run that is only partially completed because it was launched with the **launch_runs -to_step** command.

-from_step *arg* - Reset an implementation run from a specified step. This allows you to restart a run from the specified step using the **launch_runs -next_step** command.

Valid step values for ISE implementation are: NGDBuild, MAP, PAR, TRCE, XDL.

-noclean_dir - Do not clean up the run files output to the run directory. By default the tool will delete the run directory and all files within that directory when resetting the run in order to ensure a clean start when the run is reimplemented. This argument directs the tool not to remove the run directory and its content when resetting the run. In this case, when the run is reimplemented a new run directory will be created in the project runs directory.

-quiet - Execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

run - The run to reset.

Examples

The following example resets the implementation run:

```
reset_run impl_1
```

Note: The run directory and its contents will be removed from the hard disk since **-noclean_dir** is not specified.

The following example resets the synthesis run, but disables the cleanup of the run directory:

```
reset_run -noclean_dir synth_1
```

In this example, because **-noclean_dir** is specified, the synth_1 run directory is not removed and a new run directory called synth_1_2 will be created when the run is launched.

See Also

- create_run
- launch_runs

reset_ssn

Clear a SSN results set from memory

Syntax

```
reset_ssn [-quiet] name
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
name	no		Name of the set of results

Categories

Report

reset_sso

Clear a WASSO results set from memory

Syntax

```
reset_sso [-quiet] name
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
name	no		Name of the set of results

Categories

Report

reset_switching_activity

Reset switching activity on specified objects

Syntax

```
reset_switching_activity [-static_probability] [-signal_rate]
[-hier] -object_list args [-verbose] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-static_probability	yes		reset static probability
-signal_rate	yes		reset signal rate
-hier	yes		Hierarchically resets the switching activity on a hierarchical instance provided via -object_list option. This option should be used only with -object_list option
-object_list	no		objects
-verbose	yes		verbose mode
-quiet	yes		Ignore command errors

Categories

XDC, Power

Description

This command is used to reset the attributes of the switching activity on nets, ports, pins, and cells in the design.

The switching activity can be defined using the **set_switching_activity** command. The current switching activity defined for a specific port, pin, net, or cell can be found by using the **report_switching_activity** command.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-static_probability - (Optional) Reset the static probability of the specified object.

-signal_rate - (Optional) Reset the signal rate of the specified object.

-hier - (Optional) Reset the switching activity across all levels of a hierarchical object. Without **-hier**, the switching activity is applied to the specified *objects* at the current level of the hierarchy.

-verbose - (Optional) Provide verbose output from the command.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

objects - (Required) Specifies the list of objects to reset the switching activity of.

Example

The following example resets the signal_rate and static probability value on all output ports:

```
reset_switching_activity -signal_rate -static_probability [all_outputs]
```

See Also

- report_power
- reset_default_switching_activity
- set_default_switching_activity
- set_switching_activity

reset_timing

Resets the timing information on the current design

Syntax

```
reset_timing [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors

Categories

Report

reset_timing_derate

Resets constant delay derating factor

Syntax

```
reset_timing_derate [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors

Categories

SDC, XDC

reset_ucf

Clear floorplan constraints read in from a file

Syntax

```
reset_ucf [-quiet] file
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
file	no		file name

Categories

Floorplan

resize_pblock

move, resize and add and/or remove UCF ranges

Syntax

```
 resize_pblock [-add args] [-remove args] [-from args]
[-to args] [-replace] [-locs arg] [-quiet] pblock
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-add	yes		add site ranges(s)
-remove	yes		remove site ranges(s)
-from	yes		site range(s) to move
-to	yes		site range destination(s)
-replace	yes		remove all existing ranges
-locs	yes	keep_all	LOC treatment
-quiet	yes		Ignore command errors
pblock	no		Pblock to resize

Categories

Floorplan

Description

Resize, move, or remove the specified Pblock. The Pblock must first be created using the **create_pblock** command and must also be placed onto the fabric of the FPGA using the **place_pblocks** command.

A Pblock consists of a collection of slices defined by one or more independent or overlapping rectangles. Using the various options defined below, you can add slices to a rectangle, or remove slices from a rectangle, or define a new rectangle to be associated with an existing Pblock.

resize_pblock [-add <args>] [-remove <args>] [-from <args>] [-to <args>] [-replace] [-locs <arg>] [-quiet] <pblock>

Arguments

-add *args* - Add the specified range of slices to the Pblock. The slice range is specified as a rectangle from one corner to the diagonally opposite corner. For example SLICE_X0Y0:SLICE_X20Y12.

Note: A slice is the smallest area that can be specified for a Pblock. You cannot define a Pblock to cover an area smaller than a slice.

-remove *args* - Remove the specified range of slices from the Pblock. Removing slices from a Pblock may result in the Pblock being broken into multiple smaller rectangles to enforce the requirement that Pblocks are defined as one or more rectangles.

-from *args* - The **-from** and **-to** options must be used as a pair, and specify a slice or range of slices to relocate from one location to another.

-to *args* - Specifies the slice or range of slices to relocate to. Must be used with **-from**.

Note: The size of the range specified in **from** and **to** must match.

-locs *args* - Specifies how the placed logic in the Pblock will be handled as the Pblock is moved or resized. The legal values are:

keep_all - leave all locs placed as they are currently. This is the default setting when -locs is not specified. Logic that is placed outside of the Pblock will no longer be assigned to the Pblock.

keep_only_fixed - Specifies that only user-placed logic (fixed) will be preserved. Unfixed placed logic will be unplaced.

- **keep_none** - Unplace all logic.
- **move** - Specifies that all locs should be moved relative to the coordinates of the Pblock.
- **move_unfixed** - Specifies that only the unfixed placed elements should be moved. Logic placed by the user (fixed) will not be moved.
- **trim** - Specifies that logic that falls outside of the new Pblock boundaries will be unplaced. Any placed logic that still falls inside of the Pblock boundary will be left placed as it is.
- **trim_unfixed** - Trim only the unfixed placed logic.

-replace - This option specifies that the Pblock should be unplaced from the FPGA to allow it to be replaced using the **place_pblocks** command.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

pblock - Specify the Pblock to be resized, moved, or removed.

Examples

The following example places the specified Pblocks with a utilization of 75%:

```
resize_pblock block3 -add SLICE_X6Y67:SLICE_X11Y71 -remove SLICE_X6Y71:SLICE_X7Y71 -locs keep
```

The following example moves the specified Pblock by adding a range of Slices, removing the existing range of slices, and trims any placed logic. Then it adds a new range of slices to the specified Pblock in a second separate rectangle:

```
resize_pblock block3 -add SLICE_X3Y8:SLICE_X10Y3 -remove SLICE_X6Y67:SLICE_X11Y71 \
-locs trim
resize_pblock block3 -add SLICE_X6Y67:SLICE_X11Y71
```

See Also

- add_cells_to_pblock
- create_pblock
- place_pblocks

save_design

Save the current design

Syntax

```
save_design [-force] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-force	yes		Force a design save, overwriting the target UCF if necessary
-quiet	yes		Ignore command errors

Categories

Project

Description

Save any changes to the constraint files of the active constraint set. This command writes any changes to the constraint files to the project data on the hard drive; saving any work in progress and committing any changes.

Arguments

-force - Forces the PlanAhead tool to save the active constraint files regardless of whether any changes have been made.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example saves the constraint files for the active constraint set regardless of any changes to the files:

```
save_design -force
```

See Also

save_design_as

save_design_as

Save current design as a new set of constraints

Syntax

```
save_design_as [-dir arg] [-quiet] name
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-dir	yes		Directory to save design to
-quiet	yes		Ignore command errors
name	no		Name of new constraints fileset

Categories

Project

Description

Saves the active constraint set as a new constraint set, with local copies of any constraint files that are part of the constraint set. Note that the design name is not actually changed by this command.

Use this command to save changes to the constraints in a design without affecting the current constraint files. This allows you to do some "what-if" type development of design constraints.

Note: The new constraint set created by the save_design_as command will not be active in the design, although it will be referenced by the design. To make the constraint set active you must set the constrset property to point to the new constraint set for specific runs. See the example below.

Arguments

-dir *arg* - Specifies a directory to save any constraint files into. The constraint files from the active constraint set are copied into the specified directory name, with relative paths from the current constraint files preserved in the new copies.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

name - Specifies the name of the constraint set to write.

Examples

The following example saves the active constraint set into a new constraint set called newCon, and copies any constraint files for the constraint set into the specified directory:

```
save_design_as -dir C:/Data/con1 newCon
```

The following example saves the active constraint set as a new constraint set called newCon2, and copies any constraint files into the newCon2 constraint directory under project sources. The constrset property for the specified synthesis and implementation runs are then set to point to the new constraint set:

```
save_design_as newCon2
set_property constrset newCon2 [get_runs synth_1]
set_property constrset newCon2 [get_runs impl_1]
```

Note: The constraint set is not active in the design until it has been set to active for the current runs.

See Also

save_design

save_project_as

Save the current project under a new name

Syntax

```
save_project_as [-force] [-quiet] name [dir]
```

Returns

saved project object

Usage

Name	Optional	Default	Description
-force	yes		Overwrite existing project directory
-quiet	yes		Ignore command errors
name	no		New name for the project to save
dir	yes	.	Directory where the project file is saved

Categories

Project

Description

This command saves a currently open the PlanAhead tool project file under a new name in the specified directory.

Arguments

-force - This option is required to overwrite an existing project. If the project name is already define in the specified *dir* then you must also specify the **-force** option for the PlanAhead tool to overwrite the existing project.

Note: If the existing project is currently open in the PlanAhead tool, the new project will overwrite the existing project on the disk, but both projects will be opened in the PlanAhead tool. In this case you should probably run the **close_project** command prior to running **create_project**.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

name - This argument does not require a parameter name, however, it must appear before the specified *dir*. Since these commands do not have parameters, the PlanAhead tool interprets the first argument as *name* and uses the second argument as *dir*. The project file is saved as *name*.ppr and is written into the specified directory *dir*.

dir - This argument specifies the directory name to write the new project file into. If the specified directory does not exist a new directory will be created. If the directory is specified with the complete path, the PlanAhead tool uses the specified path name. However, if *dir* is specified without a path, the PlanAhead tool looks for or creates the directory in the current working directory, or the directory from which the PlanAhead tool was launched.

Note: When creating a project in GUI-mode, the PlanAhead tool appends the filename *name* to the directory name *d*ir and creates a project directory with the name *dir/name* and places the new project file and project data folder into that project directory.

Examples

The following example saves the active project as a new project called myProject in a directory called myProjectDir:

```
save_project_as myProject myProjectDir
```

Note: Because *dir* is specified as the folder name only, the PlanAhead tool will create the project in the current working directory, or the directory from which the PlanAhead tool was launched.

The following example saves the current project to a new project called myProject in a directory called C:/Designs/myProjectDir. If you use the **-force** argument, the PlanAhead tool will overwrite an existing project if one is found in the specified location.

```
save_project_as myProject C:/Designs/myProjectDir -force
```

See Also

- create_project
- open_project

select_objects

Select objects in GUI

Syntax

```
select_objects [-quiet] objects
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
objects	no		Objects to select

Categories

GUIControl

Description

This command selects the specified object in the appropriate open views in the PlanAhead tool GUI mode. This command is for display purposes only. You must use the **get_selected_objects** command to return the selected objects for use in other commands.

The **select_objects** command may select secondary objects in addition to the primary object specified. The selection of secondary objects is controlled through the use of Selection Rules defined in the **Tools > Options** command. Refer to the *PlanAhead User Guide* (ug632) for more information on Setting Selection Rules.

Selected objects can be unselected with the **unselect_objects** command.

Arguments

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

objects - Specifies one or more objects to be selected.

Example

The following example selects the specified site on the device:

```
select_objects [get_sites SLICE_X56Y214]
```

See Also

- get_selected_objects
- unselect_objects

set_case_analysis

Specify that an input is 1, 0, rising or falling

Syntax

```
set_case_analysis [-quiet] [value] objects
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
value	yes	1	Logic value on the pin: Values: 0, 1, rising, falling, zero, one, rise, fall
objects	no		List of ports or pins

Categories

SDC, XDC

Description

Specify that an input is 1, 0, rising or falling. This command is usually used to force values onto the ports and do the analysis. This is ideally suited for driving the select line of a BUFGMUX. Driving the value on the select line of BUFGMUX, will select one of the TIMESPECs for timing analysis. This prevents the analysis of unwanted TIMESPECs.

Note: this command operates silently and does not return direct feedback of its operation

Arguments

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

value - (Optional) Specifies the value that should be used on the pin for timing analysis. The valid values are 0 or zero, 1 or one, rise or rising, fall or falling. The default setting is 1.

<objects> - (Required) Specifies one or more ports or pins to apply the *value* to.

Example

This is an example set_case_analysis. The design has two clocks which are multiplexed using a BUFGMUX. Both the clocks are running at different frequencies. The default analysis of this design would be done for the clock that is defined later in the XDC. In this case, it would be CLK_B. Using set_case_analysis the analysis would be different. When (SEL = 0), the analysis would be based on CLK_A. When (SEL = 1), the analysis would be based on CLK_B.

```
create_clock -period 10.0 [get_ports CLK_A]
create_clock -period 15.0 [get_ports CLK_B]
set_case_analysis 0 [get_ports SEL]
```

See Also

report_timing

set_clock_gating_check

Capture clock-gating checks

Syntax

```
set_clock_gating_check [-setup arg] [-hold arg] [-rise] [-fall]
[-high] [-low] [-quiet] [objects]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-setup	yes	0	Clock-gating setup time: Value = 0
-hold	yes	0	Clock-gating hold time: Value = 0
-rise	yes		Specifies only rising value for the defined checks
-fall	yes		Specifies only falling value for the defined checks
-high	yes		Specifies check against high of clock waveform
-low	yes		Specifies check against low of the clock waveform
-quiet	yes		Ignore command errors
objects	yes		List of clocks, ports, pins, or cells

Categories

SDC, XDC

Description

The **set_clock_gating_check** command defines setup and hold times for gated clocks. Clock gating is a power saving technique used in many synchronous circuits. Gated clocks increase clock skew and add timing delay which must be accounted for during timing analysis.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-setup *value* - (Optional) Clock-gating setup time. Valid values are floating point numbers >= 0. The default is 0.

-hold *value* - (Optional) Clock-gating hold time. Valid values are floating point numbers >= 0. The default is 0.

-rise - (Optional) Specifies only rising value for the defined checks.

-fall - (Optional) Specifies only falling value for the defined checks.

-high - (Optional) Specifies check against high of clock waveform.

-low - (Optional) Specifies check against low of clock waveform.

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

objects - (Optional) List of clocks, ports, pins, or cells

Example

The following example will apply a setup requirement of 0.5 and a hold requirement of 0.5 on all gates in the specified clock:

```
set_clock_gating_check -setup 0.5 -hold 0.5 cpuClk
```

See Also

report_timing

set_clock_groups

Set exclusive or asynchronous clock groups

Syntax

```
 set_clock_groups [-name arg] [-logically_exclusive]
[-physically_exclusive] [-asynchronous] [-allow_paths]
[-group args] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-name	yes		Name for clock grouping
-logically_exclusive	yes		Specify logically exclusive clock groups
-physically _exclusive	yes		Specify physically exclusive clock groups
-asynchronous	yes		Specify asynchronous clock groups
-allow_paths	yes		Keep paths as constrained
-group	yes		Clocks List
-quiet	yes		Ignore command errors

Categories

SDC, XDC

Description

This command is used to define clocks, or groups of clocks, that are exclusive with or asynchronous to other clocks in the design. Exclusive or asynchronous clocks are not active at the same time, and paths between them can be ignored during timing analysis.

Using this command is similar to defining false path constraints for data paths moving between exclusive or asynchronous clock groups.

This command can also be used for multiple clocks that are derived from a single BUFGMUX as both of the clocks will not be active at the same time.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-name <*group_name*> - (Optional) Name of the clock group to be created. A name will be automatically assigned if one is not specified.

-logically_exclusive - (Optional) Indicates that the clocks listed are logically exclusive.

-physically_exclusive - (Optional) Indicates that the specified clocks are physically exclusive, and cannot exist in the design at the same time.

-asynchronous - (Optional) Specifies that the specified clocks are asynchronous to one another.

Note: **-logically_exclusive**, **-physically_exclusive** and **-asynchronous** are mutually exclusive arguments.

-allow_paths - (Optional) Preserves the previous constraints applied to the clocks included in the group.

-group <*args*> - (Optional) Species the list of clocks to be included in the clock group. Each group of clocks is exclusive with or asynchronous with the clocks specified in all other groups. If only one group of clocks is specified, that group is exclusive with or asynchronous to all other clocks in the design.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

Group all the elements driven by src_clk and sync_clk. Both the clock groups are asynchronous.

```
set_clock_groups -group [get_clocks src_clk] -group [get_clocks sync_clk] \ -asynchronous
```

See Also

set_false_path

set_clock_latency

Capture actual or predicted clock latency

Syntax

```
 set_clock_latency [-clock args] [-rise] [-fall] [-min] [-max]
[-source] [-late] [-early] [-quiet] delay objects
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-clock	yes		List of relative clocks
-rise	yes		Specify clock rise latency
-fall	yes		Specify clock fall latency
-min	yes		Specify clock rise and fall min condition latency
-max	yes		Specify clock rise and fall max condition latency
-source	yes		Specify clock rise and fall source latency
-late	yes		Specify clock rise and fall late source latency
-early	yes		Specify clock rise and fall early source latency
-quiet	yes		Ignore command errors
delay	no		Clock latency
objects	no		List of clocks, ports or pins

Categories

SDC, XDC

Description

This command defines a clock's source or network latency for specified clocks, ports, or pins.

Source latency is the time a clock signal takes to propagate from its waveform origin to the clock definition point in the design. For example, this would be the time delay for the clock to propagate from its source (oscillator) on the system board to the FPGA input port.

Network latency is the time a clock signal takes to propagate from its definition point in the design to a register clock pin on the timing path. The total clock latency at a register clock pin is the sum of a clock's source latency and network latency.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-clock *args* - (Optional) Specifies a list of clocks associated with the *latency* assigned to the specified *objects*. If the **-clock** argument is not used, the clock *latency* will be applied to all clocks passing through the specified pins and ports.

-rise - (Optional) Defines the latency for the rising clock edge.

-fall - (Optional) Defines the latency for the falling clock edge.

-min - (Optional) Defines the minimum latency for the specified clocks for multi-corner analysis.

-max - (Optional) Defines the maximum latency for the specified clocks for multi-corner analysis.

-source - (Optional) Defines the specified *latency* as a source latency. Clock source latencies can only be specified for clock objects and clock source pins.

Note: Without the **-source** argument the *latency* is considered as network latency

-late - (Optional) Indicates that the time delay specified is how late the clock edge arrives.

-early - (Optional) Indicates that the time delay specified is how early the clock edge arrives.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

latency - (Required) Specifies the amount of clock latency to apply.

objects - (Required) Specifies the clock, port, or pin objects to apply the latency to. Specifying pin or port objects assigns the latency to all register clock pins in the transitive fanout of the pins or ports. If **-clock** is used, the latency is applied to all register clock pins of the specified clocks.

Note: If *objects* specifies a clock, the **-clock** argument is unnecessary, and is ignored.

Example

This example will set an early latency on the rising edge of CLK_A.

```
set_clock_latency -source -rise -early 0.4 [get_ports CLK_A]
```

See Also

report_timing

set_clock_sense

Set clock sense on ports or pins

Syntax

```
set_clock_sense [-positive] [-negative] [-stop_propagation]
[-pulse arg] [-clocks args] [-quiet] pins
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-positive	yes		Specify positive unate (non_inverting) clock sense
-negative	yes		Specify negative unate (inverting) clock sense
-stop_propagation	yes		Stop clock propagation from specified pins
-pulse	yes		Specify pulse clock sense
-clocks	yes		List of clocks
-quiet	yes		Ignore command errors
pins	no		List of port and/or pins

Categories

SDC, XDC

Description

This is used to set clock sense at specified ports and/or pins. This is used to define the positive or negative unateness at the pin relative to a clock object. However, the specified unateness only applies at a non-unate point in the clock network, at a point where the clock signal cannot be determined. Since the clock signal is not determined, the defined clock sense propagates forward from the given pins.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-positive - (Optional) Specify positive (non_inverting) unate clock sense

-negative - (Optional) Specify negative (inverting) unate clock sense

-stop_propagation - (Optional) Stops the propagation of clocks in the **-clocks** argument from the specified pins or ports. Propagation of the clock as clock and data is stopped.

-pulse *arg* - (Optional) Specify pulse clock sense.

Note: **-positive**, **-negative**, **-stop_propagation** and **-pulse** are mutually exclusive.

-clocks *args* - (Optional) Specifies a list of clocks to apply the clock sense to for the specified pins and ports . If the **-clocks** argument is not used, the clock sense will be applied to all clocks passing through the specified pins and ports.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

pins - (Required) List of ports and/or pins to propagate the clock sense to.

Example

This command will make sure that only the positive unate paths will propagate through the output pin of the XOR gate as compared with the original clock. From this point, the tool will make sure that the positive unate is maintained to the elements the clock net is driven to.

```
set_clock_sense -positive [get_pins xor_a.z]
```

See Also

- create_clock
- get_pins

set_clock_transition

Capture predicted clock transition

Syntax

```
 set_clock_transition [-rise] [-fall] [-min] [-max]
[-quiet] transition clocks
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-rise	yes		Specify clock rising transition
-fall	yes		Specify clock falling transition
-min	yes		Specify clock rise and fall min transition
-max	yes		Specify clock rise and fall max transition
-quiet	yes		Ignore command errors
transition	no		Transition time of clock pins
clocks	no		List of clocks

Categories

SDC, XDC

Description

This command is used to define transition times on register clock pins in the fanout of the specified clocks. This is primarily used to specify the slew of the clock. Slew is the amount of time it takes for a signal to change from low to high or from high to low. This command is applicable only for ideal clocks. Once the design is implemented the clock latency can be propagated using the **set_propagated_clock** command, and the ideal clock can be ignored.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-rise - (Optional) Specify transition times for rising clock edges.

-fall - (Optional) Specify transition time for falling clock edges.

-min - (Optional) Specify clock rise and fall min transition times.

-max - (Optional) Specify clock rise and fall max transition times.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

transition - (Required) Define the transition time of the register pins clocked by the specified clocks.

clocks - (Required) Specify a list of clocks. The *transition* is applied to the register pins on the specified clocks.

Example

The following example is applied to all register clock pins in the transitive fanout of the specified clock:

```
set_clock_transition -rise 0.5 [get_clocks CLK_A]
```

See Also

- create_clock
- get_clocks
- set_propagated_clock

set_clock_uncertainty

set clock uncertainty

Syntax

```
 set_clock_uncertainty [-setup] [-hold] [-from args]
[-rise_from args] [-fall_from args] [-to args] [-rise_to args]
[-fall_to args] [-quiet] uncertainty [objects]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-setup	yes		Specify clock uncertainty for setup checks
-hold	yes		Specify clock uncertainty for hold checks
-from	yes		Specify inter-clock uncertainty source clock
-rise_from	yes		Specify inter-clock uncertainty source clock with rising edge
-fall_from	yes		Specify inter-clock uncertainty source clock with falling edge
-to	yes		Specify inter-clock uncertainty destination clock
-rise_to	yes		Specify inter-clock uncertainty destination clock with rising edge
-fall_to	yes		Specify inter-clock uncertainty destination clock with falling edge
-quiet	yes		Ignore command errors
uncertainty	no		Uncertainty of clock network
objects	yes		List of clocks, ports or pins

Categories

SDC, XDC

set_data_check

Create data to data checks

Syntax

```
set_data_check [-from args] [-to args] [-rise_from args]
[-fall_from args] [-rise_to args] [-fall_to args] [-setup]
[-hold] [-clock args] [-quiet] value
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-from	yes		From pin/port of data to data check
-to	yes		To pin/port of the data to data check
-rise_from	yes		Rise from pin/port of data to data check
-fall_from	yes		Fall from pin/port of data to data check
-rise_to	yes		Rise to pin/port of data to data check
-fall_to	yes		Fall to pin/port of data to data check
-setup	yes		Specify data check setup time
-hold	yes		Specify data check hold time
-clock	yes		Specify the clock domain at related pin/port of the checks
-quiet	yes		Ignore command errors
value	no		Setup or hold time of the defined checks

Categories

SDC, XDC

Description

Create data to data checks. This is used to do a setup and a hold check for a data pin with respect to another data pin. This is not similar to the conventional setup and hold check that is done with respect to a clock pin. So there are two parameters that are important in this constraint. One of it is the constrained pin and the other pin is the related pin. Setup and hold checks of the constrained pin are done taking the related pin as reference. (Related pin can be related to clock pin in the conventional setup and hold check)

Note: This command operates silently and does not return direct feedback of its operation

Arguments

-from *value* - (Optional) From pin/port of data to data check.

-to *value* - (Optional) To pin/port of the data to data check.

-rise_from *value* - (Optional) Rise from pin/port of data to data check.

-fall_from *value* - (Optional) Fall from pin/port of data to data check.

-rise_to *value* - (Optional) Rise to pin/port of data to data check.

-fall_to *value* - (Optional) Fall to pin/port of data to data check.

-setup *value* - (Optional) Specify setup data check setup.

-hold *value* - (Optional) Specify hold data check.

-clock *value* - (Optional) Specify the clock domain at related pin/port of the checks.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

value - (Required) Define the setup or hold time for the defined data checks.

Example

The following example defines a data check for a setup violation from pin A_IN to pin C_IN:

```
set_data_check  -from A_IN -to C_IN -setup 2.0
```

In the above example, A_IN is the related pin and C_IN is the constrained pin. The above constraint would do a setup check of C_IN with respect to A_IN. C_IN should be 2.0 ns prior to the edge of A_IN.

See Also

- remove_data_check
- report_timing

set_default_switching_activity

Set default switching activity on specified types

Syntax

```
 set_default_switching_activity [-toggle_rate arg]
[-static_probability arg] -type args [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-toggle_rate	yes	-9999.0	toggle rate value: 0% = Value = 200%. The value is % of clock.
-static_probability	yes	-9999.0	static probability value: 0 = Value = 1
-type	no		Sets the default seed values on specified types for vector-less propagation engine. List of valid default type values: input, input_set, input_reset, input_enable, register, dsp, bram_read_enable, bram_write_enable, output_enable, clock, all
-quiet	yes		Ignore command errors

Categories

XDC

Description

The switching activity of a design affects both the static and dynamic power consumption. The static power is often dependent on logic state transitions, and the dynamic power is directly proportional to the toggle rate.

The **set_default_switching_activity** command is used to specify a default activity rate for a broad class of signals when performing power estimation. The **set_switching_activity** command is used to define the activity of one or more signals, rather than the whole class.

The current default switching activity attributes can be found by using the **report_default_switching_activity** command. The values can be set to their default values by using the **reset_default_switching_activity** command.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-toggle_rate *rate* - (Optional) Specifies the toggle rate describes how often the output switches relative to the controlling clock. Valid values are between 0 and 200%. An output that switches once per clock cycle toggles at 100%. The default value is -9999.0 to indicate the argument is not in use.

-static_probability *value* - (Optional) Specifies the switching probability to be used in analysis. Valid values are 0 < value < 1. The default value is -9999.0 to indicate the argument is not in use.

Note: Either **-static_probability** or **-toggle_rate** must be specified with the **set_switching_activity** command. Both may be specified, but at least one must be specified

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

type - Specifies the type of entity for the defined switching activity. Valid values are: input, input_set, input_reset, input_enable, register, dsp, bram_read_enable, bram_write_enable, output_enable, clock, all.

Examples

The following example specifies a toggle rate of 85% for all DSP blocks:

```
set_default_switching_activity -toggle_rate 85 dsp
```

The following example specifies the toggle rate and switching probability for all supported types:

```
set_default_switching_activity -toggle_rate 19 -static_probability .22 all
```

See Also

- report_power
- reset_default_switching_activity
- reset_switching_activity
- set_switching_activity

set_delay_model

Timing Delay Model

Syntax

```
set_delay_model [-interconnect arg] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-interconnect	yes		Interconnect delay model used for timing analysis: Values: estimated, none
-quiet	yes		Ignore command errors

Categories

XDC

Description

This command is used to set the interconnect delay model for timing analysis. There are two settings for the interconnect delay model: "estimated" or "none".

If "estimated" delays are selected, the timing analysis will include an estimate of the interconnect delays based on the placement and connectivity of the design onto the device prior to implementation. If "none" is selected, then no interconnect delay is included in the timing analysis.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-interconnect *value* - (Optional) Delay model to be used. Valid values are "estimated" and "none".

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The above command will use a timing delay model which is an estimated value.

```
set_delay_model -interconnect estimated
```

See Also

report_timing

set_disable_timing

Disable timing arcs

Syntax

```
set_disable_timing [-from arg] [-to arg] [-quiet] objects
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-from	yes		From pin on cell
-to	yes		To pin on cell
-quiet	yes		Ignore command errors
objects	no		List of cells or pins, ports, lib-cells, lib-pins, libcell/cell timing-arcs

Categories

SDC, XDC

Description

This command disables timing arcs within a cell; the purpose of disabling a timing arc is to prevent timing analysis through the arc. The -from and -to arguments should either both be present or both omitted for the constraint to be valid.

Note: This command operates silently and does not return direct feedback of its operation

Arguments

-from <*pin_name*> - (Optional) Specifies the source pin of an object cell.

-to <*pin_name*> - (Optional) Specifies the destination pin of an object cell.

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

<objects> - (Required) Can be any of the following types: cells, pins, lib-cells, lib-pins, lib-cell/cell timing arcs. Timing arcs will be disabled on anything specified here.

Example

The following example disable the timing check between AX to AMUX pin of cell abc

```
set_disable_timing -from AX -to AMUX abc
```

See Also

report_timing

set_false_path

Define false path

Syntax

```
 set_false_path [-setup] [-hold] [-rise] [-fall] [-reset_path]
[-from args] [-rise_from args] [-fall_from args]
[-to args] [-rise_to args] [-fall_to args] [-through args]
[-rise_through args] [-fall_through args] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-setup	yes		Eliminate setup timing analysis for paths
-hold	yes		Eliminate hold timing analysis for paths
-rise	yes		Eliminate only rising delays for the defined paths
-fall	yes		Eliminate only falling delays for the defined paths
-reset_path	yes		Reset this path before setting false path
-from	yes		List of path startpoints or clocks
-rise_from	yes		Apply to paths rising from the list of startpoints or clocks
-fall_from	yes		Apply to paths falling from the list of startpoints or clocks
-to	yes		List of path endpoints or clocks
-rise_to	yes		Apply to paths with rise transition at the list of endpoints or clocks
-fall_to	yes		Apply to paths with fall transition at the list of endpoints or clocks
-through	yes		List of through pins, cells or nets
-rise_through	yes		Apply to paths rising through pins, cells or nets
-fall_through	yes		Apply to paths falling through pins, cells or nets
-quiet	yes		Ignore command errors

Categories

SDC, XDC

Description

This command defines false timing paths in the design that are ignored during timing analysis.

Note: This command operates silently and does not return direct feedback of its operation

Arguments

-setup - (Optional) Eliminate setup timing analysis for specified timing paths.

-hold - (Optional) Eliminate hold timing analysis for specified timing paths.

-rise - (Optional) Eliminate only rising delays for the specified timing paths.

-fall - (Optional) Eliminate only falling delays for the specified timing paths.

-reset_path - (Optional) Reset the timing path before setting false path. This clears all exception-based timing constraints from the defined timing path.

-from <*element_name*> - (Optional) List of path origins or clocks

-rise_from <*element_name*> - (Optional) Apply to paths rising from the list of origins or clocks

-fall_from <*element_name*> - (Optional) Apply to paths falling from the list of origins or clocks

-to <*element_name*> - (Optional) List of path endpoints or clocks

-rise_to <*element_name*> - (Optional) Apply to paths with rise transition at the list of endpoints or clocks

-fall_to <*element_name*> - (Optional) Apply to paths with fall transition at the list of endpoints or clocks

-through <*element_name*> - (Optional) List of through pins, cells or nets

-rise_through <*element_name*> - (Optional) Apply to paths rising through pins, cells or nets

-fall_through <*element_name*> - (Optional) Apply to paths falling through pins, cells or nets

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example eliminates the setup timing for paths from the bftClk:

```
set_false_path -setup -from bftClk
```

See Also

- get_clocks
- get_pins
- get_ports
- report_timing

set_hierarchy_separator

Set hierarchical separator character

Syntax

```
set_hierarchy_separator [-quiet] [separator]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
separator	yes	/	Hierarchy separator character

Categories

SDC, XDC

Description

This command defines the character that will be used for separating levels of hierarchy in the design.

Note: This command operates silently and does not return direct feedback of its operation

Arguments

separator - (Required) Specifies the new character to use as a hierarchy separator. The default character is '/'. Valid characters to use as the hierarchy separator are: '/', '@', '^', '#', '.', and '|'.

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

Change the hierarchy separator to the '|' character:

```
set_hierarchy_separator |
```

See Also

get_hierarchy_separator

set_ideal_latency

Specifies ideal latency

Syntax

```
set_ideal_latency [-rise] [-fall] [-min] [-max]
[-quiet] value objects
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-rise	yes		Rise ideal latency
-fall	yes		Ideal fall latency
-min	yes		Min ideal latency
-max	yes		Max ideal latency
-quiet	yes		Ignore command errors
value	no		Ideal latency value: Value = 0
objects	no		List of ports or pins

Categories

SDC, XDC

Description

Ideal networks are used for timing analysis before the clock network is propagated. This command defines the latency of ideal networks for the specified ports or pins.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-rise - (Optional) Specifies rise ideal latency.

-fall - (Optional) Specifies fall ideal latency.

Note: When neither **-rise** or **-fall** is specified, both are assumed

-min - (Optional) Specifies minimum ideal latency.

-max - (Optional) Specifies max ideal latency.

Note: When neither **-min** or **-max** is specified, both are assumed

-quiet - (Optional) Executes the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

value - (Required) Specifies the latency value for ideal networks. Latency is specified as a floating point delay value >= 0.

objects - (Required) Specifies the list of ports or pins to apply the latency to.

Examples

The following example defines the ideal latency for all inputs ports:

```
set_ideal_latency 1.5 [all_inputs]
```

See Also

set_ideal_network

set_ideal_network

Specifies an ideal network

Syntax

```
set_ideal_network [-no_propagate] [-quiet] objects
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-no_propagate	yes		Don't propagate through logic gates
-quiet	yes		Ignore command errors
objects	no		List of pins, ports or nets

Categories

SDC, XDC

Description

This command defines a network as an ideal network rather than a propagated network.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-no_propagate - (Optional) Specifies that the network should not propagate through logic gates

-quiet - (Optional) Executes the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

<objects> - (Required) Specifies the list of ports or pins to act on

Examples

The following example defines the networks from all input ports as ideal networks:

```
set_ideal_network [all_inputs]
```

See Also

set_ideal_latency

set_input_delay

Set input delay on ports or pins

Syntax

```
 set_input_delay [-clock args] [-reference_pin args]
[-clock_fall] [-level_sensitive] [-rise] [-fall]
[-max] [-min] [-add_delay] [-network_latency_included]
[-source_latency_included] [-quiet] [delay] objects
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-clock	yes		Relative clock
-reference_pin	yes		Relative pin or port
-clock_fall	yes		Delay is relative to falling edge of clock
-level_sensitive	yes		Delay is from level-sensitive latch
-rise	yes		Specifies rising delay
-fall	yes		Specifies falling delay
-max	yes		Specifies maximum delay
-min	yes		Specifies minimum delay
-add_delay	yes		Don't remove existing input delay
-network_latency _included	yes		Specifies network latency of clock already included
-source_latency _included	yes		Specifies source latency of clock already included
-quiet	yes		Ignore command errors
delay	yes	1.0	Delay value
objects	no		List of port and/or pins

Categories

SDC, XDC

Description

This command sets the input delay on ports or pins.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-clock *arg* - (Optional) Indicates that the delay is relative to the rising edge of the specified clock.

-reference_pin *arg* - (Optional) Specifies that the delay is relative to the specified pin rather than a clock.

-clock_fall - (Optional) Specifies that the delay is relative to a falling edge of the clock rather than rising edge.

-level_sensitive - (Optional) Specifies that the delay is from level-sensitive latch.

-rise - (Optional) Specifies rising delay.

-fall - (Optional) Specifies falling delay.

-max - (Optional) Specifies maximum delay.

-min - (Optional) Specifies minimum delay.

-add_delay - (Optional) Specifies that the delay specified should be added to any existing delay on the path rather than replacing the existing delay.

-network_latency_included - (Optional) Indicates that the network latency of the reference clock is included in the delay value.

-source_latency_included - (Optional) Indicates that the source latency of the relative clock is included in the specified delay.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

delay - (Optional) Specifies the delay to apply to the specified ports or pins. Valid values are floating point numbers >= 0, with a default value of 1.0.

objects - (Required) Specifies the list of ports and/or pins to assign the delay value to.

Example

The following example specifies the input delay as 3 for all ports in the design:

```
set_input_delay 3 [get_ports]
```

See Also

- get_pins
- get_ports
- report_timing
- set_output_delay

set_input_jitter

Set input jitter for a clock object

Syntax

```
set_input_jitter [-quiet] clock_name input_jitter
```

Returns

clock name

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
clock_name	no		Clock name
input_jitter	no		Input jitter: Value = 0

Categories

XDC

Description

This command sets the input jitter for a clock object.

Note: This command returns the name of the clock being operated on.

Arguments

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

<clock_name> - (Required) Specifies the clock name.

<input_jitter> - (Required, Default Value of 0) Specifies the input jitter for the specified clock object.

Example

The following example sets a input jitter of 0.7 on the specified clock:

```
set_input_jitter wbClk 0.7
```

See Also

- report_timing
- set_input_delay

set_load

Set capacitance on ports and nets

Syntax

```
set_load [-rise] [-fall] [-max] [-min] [-subtract_pin_load]
[-pin_load] [-wire_load] [-quiet] capacitance objects
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-rise	yes		Specify the rise capacitance value (for ports only)
-fall	yes		Specify the fall capacitance value (for ports only)
-max	yes		Specify the maximum capacitance value
-min	yes		Specify the minimum capacitance value
-subtract_pin_load	yes		Subtract pin capacitance from value (nets only)
-pin_load	yes		Pin Capacitance (for ports only)
-wire_load	yes		Wire Capacitance (for ports only)
-quiet	yes		Ignore command errors
capacitance	no		Capacitance value
objects	no		List of ports or nets

Categories

SDC, XDC

Description

The load capacitance is used during power analysis when running the **report_power** command, but is not used during timing analysis. This command sets the load capacitance on output ports to the specified value.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-max - (Optional) Specify the maximum load capacitance value.

-min - (Optional) Specify the minimum load capacitance value.

-rise - (Optional) Defines the rising edge load capacitance on the specified ports.

-fall - (Optional) Defines the falling edge load capacitance on the specified ports.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

capacitance - (Required) Specifies the value of the load capacitance. The value is specified as a floating point value >= 0. The default is 0.

Note: The units of the capacitance load are defined by the **set_units** command. The default unit of capacitance is picofarads (pF).

objects - (Required) Specifies a list of output port objects to assign the capacitance load to. All outputs in the design may be obtained with the **all_outputs** command.

Examples

The following example sets the specified load capacitance value for all ports:

```
set_load 5.5 [all_outputs]
```

The following example sets the rising and falling edge load capacitance for the specified output ports:

```
set_load -rise -fall 8 [get_ports wbOutput*]
```

See Also

- all_outputs
- get_ports
- report_power
- set_units

set_logic_dc

Sets logic dc for port/pins

Syntax

```
set_logic_dc [-quiet] objects
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
objects	no		List of ports or pins

Categories

SDC, XDC

Description

This command drives the specified ports or pins as "Don't Care."

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

<objects> - (Required) Specifies the list of ports or pins to be affected.

Example

The following example sets the specified port to "Don't Care":

```
set_logic_dc [get_ports reset]
```

See Also

- set_logic_one
- set_logic_zero

set_logic_one

Sets logic one for port/pins

Syntax

```
set_logic_one [-quiet] objects
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
objects	no		List of ports or pins

Categories

SDC, XDC

Description

This command sets the specified ports or pins to a logical '1'.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

<objects> - (Required) Specifies the list of ports or pins to be affected.

Example

The following example sets the specified port to logic state 1:

```
set_logic_one [get_ports reset]
```

See Also

- set_logic_dc
- set_logic_zero

set_logic_zero

Sets logic zero for port/pins

Syntax

```
set_logic_zero [-quiet] objects
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
objects	no		List of ports or pins

Categories

SDC, XDC

Description

This command sets the specified ports or pins to a logical '0'.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

<objects> - (Required) Specifies the list of ports or pins to be affected.

Example

The following example sets the specified port to logic state 0:

```
set_logic_zero [get_ports reset]
```

See Also

- set_logic_dc
- set_logic_one

set_max_delay

Specify maximum delay for timing paths

Syntax

```
 set_max_delay [-rise] [-fall] [-reset_path] [-from args]
[-rise_from args] [-fall_from args] [-to args] [-rise_to args]
[-fall_to args] [-through args] [-rise_through args]
[-fall_through args] [-quiet] delay
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-rise	yes		Delay value applies to rising path delays
-fall	yes		Delay value applies to falling path delays
-reset_path	yes		Reset this path before setting max delay
-from	yes		List of path startpoints or clocks
-rise_from	yes		Apply to paths rising from the list of startpoints or clocks
-fall_from	yes		Apply to paths falling from the list of startpoints or clocks
-to	yes		List of path endpoints or clocks
-rise_to	yes		Apply to paths with rise transition at the list of endpoints or clocks
-fall_to	yes		Apply to paths with fall transition at the list of endpoints or clocks
-through	yes		List of through pins, cells or nets
-rise_through	yes		Apply to paths rising through pins, cells or nets
-fall_through	yes		Apply to paths falling through pins, cells or nets
-quiet	yes		Ignore command errors
delay	no		Delay value

Categories

SDC, XDC

Description

Specifies the maximum delay allowed (in time units) on a timing path. The specified delay value is assigned to both the rising and falling edges of the defined timing paths unless the **-rise** or **-fall** arguments are specified.

The maximum rising and falling delay cannot be less than the minimum rising and falling delay on the same path. If this happens, the older assigned delay value is removed from the timing path and reset to 0.

The delay value is assigned to the timing path as defined by the **-from**, **-through**, and **-to** arguments. A general path delay such as **-to** endpoint will be over written by a more specific path definition such as **-from** start point **-to** endpoint, or **-from/-through/-to** path definition.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-rise - (Optional) Specifies that the delay value applies to the rising edge of the timing path.

-fall - (Optional) Specifies that the delay value applies to the falling edge of the timing path.

Note: If neither **-rise** nor **-fall** is specified, the delay is applied as both rising and falling edge path delay.

-reset_path - (Optional) Indicates that existing rising or falling edge max delays should be cleared before applying the new specified path delay. If only **-to** is specified all paths leading to the specified endpoints are cleared. If only **-from** is specified, all paths leading from the specified start points are cleared. When **-from/-to** or **-from/-through/-to** are specified, the defined paths are reset.

Note: Using this argument is the same as running **reset_paths** prior to using **set_max_delay**.

-from *value* - (Optional) Specifies the list of path start points or clocks.

-rise_from <*element_name*> - (Optional) Specifies the max delay applied to paths rising from the list of origins or clocks.

-fall_from <*element_name*> - (Optional) Specifies the max delay applied to paths falling from the list of origins or clocks.

-to <*element_name*> - (Optional) Specifies the list of path endpoints or clocks.

-rise_to <*element_name*> - (Optional) Specifies the max delay applied to paths with rise transition at the list of endpoints or clocks.

-fall_to <*element_name*> - (Optional) Specifies the max delay applied to paths with fall transition at the list of endpoints or clocks.

-through <*element_name*> - (Optional) Specifies the list of through pins, cells or nets.

-rise_through <*element_name*> - (Optional) Specifies the max delay applied to paths rising through pins, cells or nets.

-fall_through <*element_name*> - (Optional) Specifies the max delay applied to paths falling through pins, cells or nets.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

delay - (Required) Specifies the maximum delay value in units of time. The *delay* can be specified as a floating point number >= 0, with a default value of 0.

Note: The units of time are defined by the **set_units** command. The default unit of time is nanoseconds (ns).

Examples

The following example defines a global (design-wide) maximum delay of 20 time units:

```
set_max_delay 20
```

The following example clears the existing max delay and specifies a new maximum delay for paths to endpoints clocked by the specified clock:

```
set_max_delay -reset_path 6 -to [get_clocks cpuClk]
```

See Also

- get_clocks
- report_timing
- set_min_delay
- set_units

set_max_fanout

Set maximum fanout for ports or cells

Syntax

```
set_max_fanout [-quiet] [fanout] ports
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
fanout	yes	0.0	Max Fanout
ports	no		List of ports

Categories

SDC, XDC

Description

This command specifies the maximum fanout allowed for the port or ports specified in the *ports* argument. Altering the fanout allowed affects but placement and routing as well timing closure.

While specifying fanout is most common for clock nets it is valid for any driver net in the design.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-quiet - (Optional) Executes the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

fanout - (Optional, Default Value of 0.0) Specifies that maximum fanout allowed on the port or ports specified by *ports*.

ports - Specifies a list of port objects that the maximum fanout specification should be applied to.

Examples

Specify that the net named "clk" in the top level of hierarchy is allowed to have a maximum fanout of 12 loads

```
set_max_fanount 12 top\clk
```

Specify that all of the clocks that have a period of 10ns should have a maximum fanout of 200

```
set_max_fanout 200 [get_clocks * "period = 10"]
```

See Also

- get_clocks
- get_pins
- get_ports
- get_nets

set_max_time_borrow

Limit time borrowing for latches

Syntax

```
set_max_time_borrow [-quiet] delay objects
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
delay	no		Delay value for borrowing: Value = 0
objects	no		List of clocks, cells, data pins or clock pins

Categories

SDC, XDC

Description

This command specifies the maximum amount of time that can be borrowed between nets when analyzing the timing on latches

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-quiet - (Optional) Executes the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

delay - (Required) Specifies the delay (in time units) that should be applied to the pins specified in the *objects* field. The *delay* can be specified as a floating point number >= 0, with a default value of 0.

Note: The units of time are defined by the **set_units** command. The default unit of time is nanoseconds (ns).

objects - (Required) Specifies a list of clocks, cells, data pins, or clock pins to which the limit should be applied.

Examples

Specifies that the latches attached to "all clocks" will be allowed 0 time units of borrowing. Effectively, this disables time borrowing throughout the entire design.

```
set_max_time_borrow 0.0 [all_clocks]
```

Specifies that nets in the top level of hierarchy are allowed 20 time units of time borrowing.

```
set_max_time_borrow 20 {top/*}
```

See Also

- all_clocks
- get_clocks
- get_nets

set_min_delay

Specify minimum delay for timing paths

Syntax

```
 set_min_delay [-rise] [-fall] [-reset_path] [-from args]
[-rise_from args] [-fall_from args] [-to args] [-rise_to args]
[-fall_to args] [-through args] [-rise_through args]
[-fall_through args] [-quiet] delay
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-rise	yes		Delay value applies to rising path delays
-fall	yes		Delay value applies to falling path delays
-reset_path	yes		Reset this path before setting min delay
-from	yes		List of path startpoints or clocks
-rise_from	yes		Apply to paths rising from the list of startpoints or clocks
-fall_from	yes		Apply to paths falling from the list of startpoints or clocks
-to	yes		List of path endpoints or clocks
-rise_to	yes		Apply to paths with rise transition at the list of endpoints or clocks
-fall_to	yes		Apply to paths with fall transition at the list of endpoints or clocks
-through	yes		List of through pins, cells or nets
-rise_through	yes		Apply to paths rising through pins, cells or nets
-fall_through	yes		Apply to paths falling through pins, cells or nets
-quiet	yes		Ignore command errors
delay	no		Delay value

Categories

SDC, XDC

Description

Specifies the minimum delay allowed (in time units) on a timing path. The specified delay value is assigned to both the rising and falling edges of the defined timing paths unless the **-rise** or **-fall** arguments are specified.

The minimum rising and falling delay cannot be greater than the maximum rising and falling delay on the same path. If this happens, the older assigned delay value is removed from the timing path and reset to 0.

The delay value is assigned to the timing path as defined by the **-from**, **-through**, and **-to** arguments. A general path delay such as **-to** endpoint will be over written by a more specific path definition such as **-from** start point **-to** endpoint, or **-from/-through/-to** path definition.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-rise - (Optional) Specifies that the delay value applies to the rising edge of the timing path.

-fall - (Optional) Specifies that the delay value applies to the falling edge of the timing path.

-reset_path - (Optional) Indicates that existing rising or falling edge min delays should be cleared before applying the new specified path delay. If only **-to** is specified all paths leading to the specified endpoints are cleared. If only **-from** is specified, all paths leading from the specified starting points are cleared. When **-from/-to** or **-from/-through/-to** are specified, the defined paths are reset.

Note: Using this argument is the same as running **reset_paths** prior to using **set_min_delay**.

-from *objects* - (Optional) Defines the starting points of the timing paths that will be assigned the specified delay.

-rise_from *objects* - (Optional) Specifies the starting points of the timing path that will have the specified delay assigned to its rising edge.

-fall_from *objects* - (Optional) Specifies the starting points of the timing path that will have the specified delay assigned to its falling edge.

-to *objects* - (Optional) Specifies the destination objects for the path that will be affected by the minimum delay

-rise_to *objects* - (Optional) Specifies the destination objects for the rising-edge path that will be affected by the minimum delay

-fall_to *objects* - (Optional) Specifies the destination objects for the falling-edge path that will be affected by the minimum delay

-through *objects* - (Optional) Specifies the list of pins, cell, or nets that the path affected by the minimum delay travels through.

-rise_through *objects* - (Optional) Specifies the list of pins, cell, or nets that the rising-edge path affected by the minimum delay travels through.

-fall_through *objects* - (Optional) Specifies the list of pins, cell, or nets that the falling-edge path affected by the minimum delay travels through.

-quiet - (Optional) Executes the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

delay - (Required) Specifies the minimum delay value in units of time. The *delay* can be specified as a floating point number >= 0, with a default value of 0.

Note: The units of time are defined by the **set_units** command. The default unit of time is nanoseconds (ns).

Examples

The following example specifies a global (design-wide) minimum delay of 20 time units:

```
set_min_delay 20
```

The following example defines a minimum delay of 20 for timing paths with endpoints at all output ports:

```
set_min_delay 20 -to [get_ports -filter {DIRECTION == out}]
```

See Also

- get_clocks
- get_nets
- get_ports
- report_timing
- set_units

set_msg_limit

Set message limit, return message limit

Syntax

```
set_msg_limit [-severity arg] [-id arg] [-quiet] count
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-severity	yes	ALL	Message severity to be set (not valid with -id,) e.g. "ERROR" or "CRITICAL WARNING"
-id	yes		Unique message id to be set (not valid with -severity,) e.g Common-99
-quiet	yes		Ignore command errors
count	no		New message limit

Categories

Report

Description

This command defines the number of messages that will be returned by the PlanAhead application during a design session, or single invocation. You can report the current message limit of a message type or ID with the **get_msg_limit** command.

The limit of a specific message ID or severity of message can be defined, or a limit for all messages returned by the PlanAhead application can be defined. For instance the following are two messages returned under different circumstances:

INFO: [common-99] This is an example INFO message

CRITICAL WARNING: [Netlist-1129] This message is a CRITICAL WARNING and requires user attention

When a message ID or message severity has not been specified, this command will set the message limit for all messages returned by the PlanAhead application.

Note: You can change the severity of a specific message ID with the **set_msg_severity** command. You can restore the original message limit using the **reset_msg_limit** command.

Arguments

-id *value* - is found in the PlanAhead application in the Messages view or other reports when the message is reported. Use the specific message ID for the message of interest for use in this command. For example, the message IDs above are common-99 and Netlist-1129.

-severity *value* - Specifies the severity of the message. There are five message severities:

- **ERROR** - An ERROR condition implies an issue has been encountered which will render design results unusable and cannot be resolved without user intervention.
- **{CRITICAL WARNING}** - A CRITICAL WARNING message indicates that certain input/constraints will either not be applied or are outside the best practices for a FPGA family. User action is strongly recommended. Note: Since this is a two word value, it must be enclosed in {}.
- **WARNING** - A WARNING message indicates that design results may be sub-optimal because constraints or specifications may not be applied as intended. User action may be taken or may be reserved.
- **INFO** - An INFO message is the same as a STATUS message, but includes a severity and message ID tag. An INFO message includes a message ID to allow further investigation through answer records if needed.
- **STATUS** - A STATUS message communicates general status of the process and feedback to the user regarding design processing. A STATUS message does not include a message ID.

-quiet - This option tells the PlanAhead application to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

count - Specifies the message limit for a specific message or type of message. This is a required argument. You can specify the maximum message limit with a count of -1.

Examples

The following example sets the message limit to 50 for all message severities:

```
set_msg_limit 50
```

Note: While this limit may be appropriate for a specific message ID, this is a very small limit for all messages.

The following example sets the message limit of the specified message ID:

```
set_msg_limit -id Netlist-1129 100
```

The following example sets the maximum message limit for all messages:

```
set_msg_limit -1
reset_msg_limit
```

Note: Both lines restore the default, maximum message limit.

See Also

- get_msg_limit
- reset_msg_limit
- set_msg_severity

set_msg_severity

Set Message Severity by ID

Syntax

```
set_msg_severity [-quiet] id severity
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
id	no		Unique message id to be set, e.g. "Common-99"
severity	no		Message severity to be changed to, e.g. "ERROR" or "CRITICAL WARNING"

Categories

Project

Description

Changes the severity of a specified message ID from one type, such as WARNING, to another type, such as ERROR.

Use this command to customize the message severity returned by the PlanAhead application to specific levels appropriate to your usage. For instance the following are two messages returned by the PlanAhead application under different circumstances:

- INFO: [common-99] This is an example INFO message
- CRITICAL WARNING: [Netlist-1129] This message is a CRITICAL WARNING and requires user attention

The message IDs above are common-99 and Netlist-1129. You can change the severity of these message IDs from INFO to WARNING, or from CRITICAL WARNING to ERROR if desired.

Note: You can restore the message severity of a specific message ID to its original setting with the **reset_message_severity** command.

Arguments

id - is found in the PlanAhead application in the Messages view or other reports when the message is reported. Use the specific message ID for the message of interest for use in this command.

severity - Specifies the severity of the message. There are five message severities:

- **ERROR** - An ERROR condition implies an issue has been encountered which will render design results unusable and cannot be resolved without user intervention.
- **{CRITICAL WARNING}** - A CRITICAL WARNING message indicates that certain input/constraints will either not be applied or are outside the best practices for a FPGA family. User action is strongly recommended. Note: Since this is a two word value, it must be enclosed in {}.
- **WARNING** - A WARNING message indicates that design results may be sub-optimal because constraints or specifications may not be applied as intended. User action may be taken or may be reserved.
- **INFO** - An INFO message is the same as a STATUS message, but includes a severity and message ID tag. An INFO message includes a message ID to allow further investigation through answer records if needed.
- **STATUS** - A STATUS message communicates general status of the process and feedback to the user regarding design processing. A STATUS message does not include a message ID. Note: Because STATUS messages do not have message IDs, you cannot change the severity level of a STATUS message.

-quiet - This option tells the PlanAhead application to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example changes the message ID **common-99** from an INFO message to a WARNING:

```
set_msg_severity common-99 WARNING
```

This elevates the standard INFO message to a WARNING so it demands greater attention when encountered.

The following example changes the message ID **Netlist-1129** from a CRITICAL WARNING to a WARNING:

```
set_msg_severity Netlist-1129 WARNING
```

This reduces the significance of this CRITICAL WARNING to a simple WARNING so that it may cause less concern when encountered.

See Also

- get_msg_limit
- reset_msg_severity
- set_msg_limit

set_multicycle_path

Define multicycle path

Syntax

```
 set_multicycle_path [-setup] [-hold] [-rise] [-fall]
[-start] [-end] [-reset_path] [-from args] [-rise_from args]
[-fall_from args] [-to args] [-rise_to args] [-fall_to args]
[-through args] [-rise_through args] [-fall_through args]
[-quiet] [path_multiplier ]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-setup	yes		Only setup multiplier is set
-hold	yes		Only hold multiplier is set
-rise	yes		Multiplier valid for rising delays on path endpoint
-fall	yes		Multiplier valid for falling delays on path endpoint
-start	yes		Multiplier measured against path startpoint
-end	yes		Multiplier measured against path endpoint
-reset_path	yes		Reset this path before setting multicycle
-from	yes		List of path startpoints or clocks
-rise_from	yes		Apply to paths rising from the list of startpoints or clocks
-fall_from	yes		Apply to paths falling from the list of startpoints or clocks
-to	yes		List of path endpoints or clocks
-rise_to	yes		Apply to paths with rise transition at the list of endpoints or clocks
-fall_to	yes		Apply to paths with fall transition at the list of endpoints or clocks
-through	yes		List of through pins, cells or nets
-rise_through	yes		Apply to paths rising through pins, cells or nets
-fall_through	yes		Apply to paths falling through pins, cells or nets

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
path_multiplier	yes	1	Eliminate setup timing analysis for paths

Categories

SDC, XDC

Description

This command allows the user to specify the total number of clock cycles required for propagation of a signal from its origin to destination when that propagation is longer than a single clock cycle.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-setup - (Optional) Specifies that the multiplier should be applied to setup time calculations

-hold - (Optional) Specifies that the multiplier should be applied to hold time calculations

-rise - (Optional) Specifies that the multiplier should be applied to rising edge delays on the path endpoint

-fall - (Optional) Specifies that the multiplier should be applied to falling edge delays on the path endpoint

-start - (Optional) Specifies that the multiplier should be measured against the path origin

-end - (Optional) Specifies that the multiplier should be measured against the path endpoint

-reset_path - (Optional) Specifies that the path specified should be reset before applying the multicycle path multiplier to it

-from *objects* - (Optional) Specifies the origin objects for the path that will be affected by the multicycle path multiplier.

-rise_from *objects* - (Optional) Specifies the origin objects for the rising-edge path that will be affected by the multicycle path multiplier.

-fall_from *objects* - (Optional) Specifies the origin objects for the falling-edge path that will be affected by the multicycle path multiplier.

-to *objects* - (Optional) Specifies the destination objects for the path that will be affected by the multicycle path multiplier.

-rise_to *objects* - (Optional) Specifies the destination objects for the rising-edge path that will be affected by the multicycle path multiplier.

-fall_to *objects* - (Optional) Specifies the destination objects for the falling-edge path that will be affected by the multicycle path multiplier.

-through *objects* - (Optional) Specifies the list of pins, cell, or nets that the path affected by the multicycle path multiplier travels through.

-rise_through *objects* - (Optional) Specifies the list of pins, cell, or nets that the rising-edge path affected by the multicycle path multiplier travels through.

-fall_through *objects* - (Optional) Specifies the list of pins, cell, or nets that the falling-edge path affected by the multicycle path multiplier travels through.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

path_multiplier - (Optional, Default Value of 1) Specifies the number of clock cycles required for the path specified

Examples

Specify a global multicycle path delay of 5

```
set_multicycle_path 5
```

Set a multicycle path from the CPUs in the design to the RAMs in the design

```
set_multicycle_path -from [all_cpus] -to [all_rams] 3
```

See Also

- get_nets
- get_pins
- get_clocks

set_operating_conditions

Set operating conditions for power estimation

Syntax

```
 set_operating_conditions [-voltage args] [-grade arg]
[-process arg] [-junction_temp arg] [-ambient_temp arg]
[-thetaja arg] [-thetasa arg] [-airflow arg] [-heatsink arg]
[-thetajb arg] [-board arg] [-board_temp arg]
[-board_layers arg] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-voltage	yes		List of voltage pairs, e.g., {name value}. Supported voltage supplies vary by family.
-grade	yes	commercial	Temperature grade. Supported values vary by family.
-process	yes	typical	Process data: typical or maximum
-junction_temp	yes	auto	Junction Temperature (C): auto|degC
-ambient_temp	yes	default	Ambient Temperature (C): default|degC
-thetaja	yes	auto	ThetaJA (C/W): auto|degC/W
-thetasa	yes	auto	ThetaSA (C/W): auto|degC/W
-airflow	yes	varies by family	Airflow (LFM): 0 to 750
-heatsink	yes	medium	Dimensions of heatsink: none, low, medium, high, custom
-thetajb	yes	auto	ThetaJB (C/W): auto|degC/W
-board	yes	medium	Board type: jedec, small, medium, large, custom
-board_temp	yes		Board Temperature degC
-board_layers	yes	8to11	Board layers: 4to7, 8to11, 12to15, 16+
-quiet	yes		Ignore command errors

Categories

SDC, XDC

Description

The environmental operating conditions of the device are used for power analysis when running the **report_power** command, but are not used during timing analysis. This command specifies the real-world operating conditions that are used when performing analysis of the design.

Operating conditions can be restored to their default values with the use of the **reset_operating_conditions** command. Current operating conditions can be reported with the **report_operating_conditions** command.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-voltage *arg* - (Optional) Specifies the list of voltage pairs. Supported voltage supplies vary by family.

-grade *arg* - (Optional) Specifies the temperature grade of the target device. Supported values vary by family. The default value is "commercial".

-process *arg* - (Optional) Specifies the manufacturing process characteristics to be assumed. The valid values are "typical" or "maximum". The default value is "typical".

-junction_temp *arg* - (Optional) Specifies the device junction temperature used for modeling. Valid values are "auto" or an actual temperature specified in degrees C. The default value is "auto".

-ambient_temp *arg* - (Optional) Specifies the environment ambient temperature in degrees C. The default setting is "default".

-thetaja *arg* - (Optional) Specifies the Theta-JA thermal resistance used for modeling in degrees C/W. The default setting is "auto".

-thetasa *arg* - (Optional) Specifies the Theta-SA thermal resistance used during modeling in degrees C/W. The default setting is "auto".

-airflow *[0:750]* - (Optional) Specifies the Linear Feet Per Minute (LFM) airflow to be used for modeling. The default setting varies by device family.

-heatsink *arg* - (Optional) Specifies the heatsink profile to be used during modeling. Valid values are: none, low, medium, high, custom. The default setting is "medium".

-thetajb *arg* - (Optional) Specifies the Theta-JB thermal resistance used for modeling in degrees C/W. The default setting is "auto".

-board *arg* - (Optional) Specifies the board size to be used for modeling. The valid values are: jedec, small, medium, large, custom. The default value is "medium".

-board_temp arg - (Optional) Specifies the board temperature in degrees Centigrade to be used for modeling.

-board_layers *arg* - (Optional) Specifies the number of board layers to be used for modeling. Valid values are: "4to7" for boards with 4 to 7 layers, "8to11" for boards with 8 to 11 layers, "12to15" for boards with 12 to 15 layers, and "16+" for boards with 16 or more layers. The default setting is "8to11".

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example specifies an industrial grade device with an ambient operating temperature of 75 degrees C

```
set_operating_conditions -grade industrial -ambient_temp 75
```

See Also

reset_operating_conditions

set_output_delay

Set output delay on ports or pins

Syntax

```
 set_output_delay [-clock args] [-reference_pin args]
[-clock_fall] [-level_sensitive] [-rise] [-fall]
[-max] [-min] [-add_delay] [-network_latency_included]
[-source_latency_included] [-quiet] [delay] objects
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-clock	yes		Relative clock
-reference_pin	yes		Relative pin or port
-clock_fall	yes		Delay is relative to falling edge of clock
-level_sensitive	yes		Delay is from level-sensitive latch
-rise	yes		Specifies rising delay
-fall	yes		Specifies falling delay
-max	yes		Specifies maximum delay
-min	yes		Specifies minimum delay
-add_delay	yes		Don't remove existing input delay
-network_latency _included	yes		Specifies network latency of clock already included
-source_latency _included	yes		Specifies source latency of clock already included
-quiet	yes		Ignore command errors
delay	yes	1.0	Delay value
objects	no		List of port and/or pins

Categories

SDC, XDC

Description

This command sets the output delays on specified ports.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-clock *arg* - (Optional) Indicates that the delay is relative to the rising edge of the specified clock.

-reference_pin *arg* - (Optional) Specifies that the delay is relative to the specified pin rather than a clock.

-clock_fall - (Optional) Specifies that the delay is relative to a falling edge of the clock rather than rising edge.

-level_sensitive - (Optional) Specifies that the delay is sourced by a level-sensitive latch

-rise - (Optional) Specifies that the delay is for a rising edge.

-fall - (Optional) Specifies that the delay is for a falling edge

-max - (Optional) Specifies that the delay specified should be treated as a maximum threshold.

-min - (Optional) Specifies that the delay specified should be treated as a minimum threshold.

-add_delay - (Optional) Specifies that the delay specified should be added to any existing delay on the path rather than replacing the existing delay.

-network_latency_included - (Optional) Specifies that the network latency of the reference clock is included in the specified delay value.

-source_latency_included - (Optional) Specifies that the source latency of the reference clock is included in the specified delay value.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

delay - (Optional) Specifies the delay to apply to the specified ports or pins. Valid values are floating point numbers >= 0, with a default value of 1.0.

objects - (Required) Specifies the list of ports and/or pins to assign the delay value to.

Examples

The following example sets an output delay on ports relative to the specified clock:

```
set_output_delay 5.0 -clock [get_clocks cpuClk] [get_ports]
```

The next example is the same as the prior example except that network latency is now included:

```
set_output_delay 5.0 -clock [get_clocks cpuClk] -network_latency_included [get_ports]
```

See Also

- get_ports
- set_input_delay

set_package_pin_val

Set user columns on one or more package pins

Syntax

```
set_package_pin_val -package_pins args -column arg -value arg [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-package_pins	no		Package pin names
-column	no		User column name
-value	no		Value to set
-quiet	yes		Ignore command errors

Categories

PinPlanning

set_param

Set a parameter value

Syntax

```
set_param [-quiet] name value
```

Returns

newly set parameter value

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
name	no		Parameter name
value	no		Parameter value

Categories

PropertyAndParameter

Description

Defines the value of a user-definable configuration parameter within the PlanAhead tool. These parameters configure and control various behaviors of the PlanAhead tool. Refer to **report_param** for a description of the currently defined parameters.

You can use the **reset_param** command to restore any parameter that has been modified back to its default setting.

Note: Setting a specified parameter value to -1 will disable the feature in the PlanAhead tool.

Arguments

name - Specifies the name of the parameter to set the value of. You can only set the value of one parameter at a time.

value - Specifies the value to set the specified parameter to.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example sets the value of the specified parameter:

```
set_param tcl.statsThreshold 15
```

See Also

- get_param
- list_param
- report_param
- reset_param

set_propagated_clock

Specify propagated clock latency

Syntax

```
set_propagated_clock [-quiet] objects
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
objects	no		List of clocks, ports, or pins

Categories

SDC, XDC

Description

This command propagates clock latency throughout a clock network. This results in more accurate skew and timing results throughout the clock network.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

objects - (Required) Specifies the clock objects to force propagation on.

Examples

Specify that the primary system clock from the top-level should be propagated

```
set_propagated_clock [get_clocks top/clk]
```

Specify that all clocks from "sublevel1" should be propagated

```
set_propagated_clock [get_clocks sublevel1/*]
```

See Also

- get_clocks
- create_clock

set_property

Set property on object(s)

Syntax

```
set_property [-quiet] property_name property_value objects ...
```

Returns

The value that was set if success, "" if failure

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
property_name	no		Name of property to set
property_value	no		Value of property to set
objects	no		Objects to set properties

Categories

XDC, Object, PartialReconfiguration, PropertyAndParameter, Partition

Description

Assigns the defined property *name* and *value* to the specified *objects*.

The **set_property** command can be used to configure the properties of a synthesis or implementation run. The run can be created with the use of the **create_run** command, and can be launched with **launch_runs**. The examples below show properties being defined for both an XST run and a synthesis run.

This command can be used to define any property on an object in the design. Each object has a set of predefined properties that have expected values, or a range of values. The set_property command can be used to define the values for these properties. To determine the defined set of properties for an object, use the **report_property**, **list_property**, or **list_property_values** commands to display the properties on an object.

You can also define custom properties for an object, by specifying a unique *name* and *value* pair for the object. If an object has custom properties, these will also be reported by the **report_property** and **list_property** commands.

One use of this command is to configure the properties of a synthesis or implementation run. The run can be created with the use of the **create_run** command, and can be launched with **launch_runs**. The examples below show properties being defined for an XST run synthesis run.

This command defines the value of a single attribute of a specified synthesis or implementation program. You will need multiple **set_property** commands to define all of the various options needed to configure XST for synthesis, or NGDBUILD, MAP, PAR, and TRACE for implementation.

Note You can use the **report_property** command to list the properties and values on a synthesis or implementation run that can be set.

Refer to the *XST User Guide (ug627)* and the *Command-Line Tools User Guide (ug628),* for more information on each of these tools and the various required and optional settings that can be configured.

Arguments

-quiet - Execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

name - The name of the property to be assigned to the object or objects.

Note: *name* is case sensitive and should be specified appropriately

value - The value to assign to the *name* on the specified object or objects. The value is checked against the property type to ensure that the value is valid. If the value is not appropriate for the property an error will be returned.

objects - One or more objects to assign the property to.

Examples

The following example sets a property called TRUTH to false on the specified cell:

```
set_property TRUTH false [lindex [get_cells] 1]
```

The following example configures properties of the synth_1 run, setting options for XST:

```
set_property flow {XST 13} [get_runs synth_1]
set_property strategy {PlanAhead Defaults} [get_runs synth_1]
set_property args.xst.opt_level 2 [get_runs synth_1]
set_property args.xst.fsm_encoding one-hot [get_runs synth_1]
set_property args.xst.use_dsp48 automax [get_runs synth_1]
```

The following example sets the internal vref property value for the specified IO Bank:

```
set_property internal_vref {0.75} [get_iobanks 0]
```

The following example defines a DCI Cascade by setting the slave_banks property for the specified IO Bank:

```
set_property slave_banks  {14} [get_iobanks 0 ]
```

See Also

- create_property
- create_run
- get_cells
- get_property
- get_runs
- launch_runs
- list_property
- list_property_value
- report_property
- reset_property

set_speed_grade

Timing Speed Grade

Syntax

```
set_speed_grade [-quiet] value
```

Returns

string result

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
value	no		Speed grade used for timing analysis

Categories

Project

Description

Set the speed grade for the target device in the current design. This command is used to change the speed grade of the target device for timing analysis. It must be run on an opened Synthesized or Implemented Design. It is usually run prior to the report_timing command or other timing commands to change the speed grade for analysis.

Arguments

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

value - Specifies the speed grade for the target device. Legal values are -1, -2, or -3.

Example

The following example sets the speed grade for the device in the current design to -1:

```
set_speed_grade -1
```

See Also

set_property

set_switching_activity

Set switching activity on specified objects or default types

Syntax

```
 set_switching_activity [-toggle_rate arg] [-type args]
[-static_probability arg] [-signal_rate arg] [-hier]
[-object_list args] [-verbose] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-toggle_rate	yes	-9999.0	toggle rate value: 0% = Value = 100%. Use with RTL power estimation.
-type	yes		Use with RTL power estimation. List of valid type values: registers, inputs, outputs, inouts, ports, outputEnable, three_states, dsps, brams, bramWrite, bramEnable, clockEnable
-static_probability	yes	-9999.0	static probability value: 0 = Value = 1
-signal_rate	yes	-9999.0	signal rate
-hier	yes		Hierarchically sets the switching activity on a hierarchical instance provided via -object_list option. This option should be used only with -object_list option
-object_list	yes		objects
-verbose	yes		verbose mode
-quiet	yes		Ignore command errors

Categories

XDC, Power

Description

The switching activity of a design affects both the static and dynamic power consumption. The static power is often dependent on logic state transitions, and the dynamic power is directly proportional to the toggle rate.

This command specifies the signal rate and/or the switching probability to be used when performing power estimation.

The **set_default_switching_activity** command is used to specify a default activity rate for a broad class of signals when performing power estimation. The **set_switching_activity** command is used to define the activity of one or more signals, rather than the whole class.

The current switching activity attributes can be found by using the **report_switching_activity** command. The values can be set to their default values by using the **reset_switching_activity** command.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-toggle_rate *rate* - (Optional) Specifies the toggle rate describes how often the output switches relative to the controlling clock. Valid values are between 0 and 200%. An output that switches once per clock cycle toggles at 100%. The default value is -9999.0 to indicate the argument is not in use.

-type *value* - (Optional)Specifies the type of logic entity for the defined switching activity. Valid types are: registers, inputs, outputs, inouts, ports, outputEnable, three_states, dsps, brams, bramWrite, bramEnable, clockEnable.

-static_probability *value* - (Optional) Specifies the switching probability to be used in analysis. Valid values are 0 < value < 1. The default value is -9999.0.

-signal_rate *value* - (Optional) Specifies the signal frequency to be used for analysis. The default value is -9999.0 to indicate the argument is not in use.

Note: Either **-static_probability** or **-signal_rate** must be specified with the **set_switching_activity** command. Both may be specified, but at least one must be specified

-hier - (Optional) Specifies the switching activity should be applied to signals at all levels of a hierarchical instance. Without **-hier**, the switching activity is applied to the specified *objects* at the current level of the hierarchy.

-verbose - (Optional) Specifies that the command should be verbose about its activities.

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

objects - (Optional) Specifies a list of objects to which the switching activity configuration should be applied.

Example

The following example specifies a signal rate and switching probability for all ports, then reports the switching attributes for the ports:

```
set_switching_activity -signal_rate 55 -static_probability .27 [get_ports]
report_switching_activity [get_ports]
```

See Also

- report_power
- reset_default_switching_activity
- reset_switching_activity
- set_default_switching_activity

set_system_jitter

Set system jitter

Syntax

```
set_system_jitter [-quiet] system_jitter
```

Returns

system_jitter

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
system_jitter	no		System jitter: Value = 0

Categories

XDC

Description

This command specifies the system-wide jitter (in time units) to be used for timing analysis.

Note: This command returns the value of the system jitter that was set.

Arguments

-quiet - (Optional) Executes the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

jitter - (Required, Default Value of 0) Specifies the system jitter (in time units) to be applied system-wide

Example

Set a system-wide jitter of 3 time units

```
set_system_jitter 3
```

See Also

- set_input_jitter
- set_input_delay
- set_clock_uncertainty

set_timing_derate

Specify constant delay derating factor

Syntax

```
set_timing_derate [-early] [-late] [-clock] [-data] [-net_delay]
[-cell_delay] [-cell_check] [-quiet] derate objects
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-early	yes		Specify early derate factor
-late	yes		Specify late derate factor
-clock	yes		Specify derate factor for clock paths only
-data	yes		Specify derate factor for data paths only
-net_delay	yes		Specify derate factor for net delays only
-cell_delay	yes		Specify derate factor for cell delays only
-cell_check	yes		Specify derate factor for cell timing checks only
-quiet	yes		Ignore command errors
derate	no		Derate factor: Value = 0
objects	no		List of cells, library cells or nets

Categories

SDC, XDC

Description

This command specifies the derating factor to be used when performing timing analysis. Timing delays calculated during timing analysis are multiplied by the specified derating factor.

You can set the derating factor for the design, and then you can specify objects to set specific derating values for cells, clocks, or nets in the design. To set derating values for the design simply issue the command with no *objects* specified.

You can restore the derating factor for the design using the **reset_timing_derate** command.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-early - (Optional) Specifies the derating factor for shortest paths.

-late - (Optional) Specifies the derating factor for longest paths.

-clock - (Optional) Defines the derating factor applied to clock paths only.

-data - (Optional) Defines the derating factor applied to data paths only.

-net_delay - (Optional) Specifies that the derating factor is to be applied to net delays only.

-cell_delay - (Optional) Specifies that the derating factor is to be applied for cell delay checks only.

-cell_check - (Optional) Specifies that the derating factor is to be applied for cell timing checks only.

-quiet - (Optional) Executes the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

derate - (Required) Specifies the derating factor to be applied. The default value is 1.0. The derating factor can be specified as a floating point number between 0 and 1. A derating factor of 0.9 derates performance by 10%.

objects - (Required) Specifies the objects that to be derated. If no *objects* are specified, the derating factor is applied globally to the design.

Examples

Derate the performance of High-Speed IO clock paths by 20%:

```
set_timing_derate -clock 0.8 [all_hsios]
```

See Also

- get_clocks
- get_nets
- get_pins
- get_ports
- get_cells
- report_timing
- reset_timing_derate

set_units

Set units for checking

Syntax

```
set_units [-capacitance arg] [-time arg] [-current arg]
[-voltage arg] [-power arg] [-resistance arg] [-suffix arg]
[-digits arg] [-quiet]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-capacitance	yes	pF	Capacitance unit in farad. Valid values are from kF-fF.
-time	yes	ns	Time unit in seconds. Valid values are from ks-fs.
-current	yes	mA	Current unit in ampere. Valid values are from kA-fA.
-voltage	yes	V	Voltage unit in volt. Valid values are from kV-fV.
-power	yes	mW	Wattage unit in watts. Valid values are from kW-fW.
-resistance	yes	ohm	Resistance unit in ohm. Valid values are from kOhm-fOhm.
-suffix	yes		Suffix for units
-digits	yes	1	Number of digits
-quiet	yes		Ignore command errors

Categories

SDC, XDC

Description

This command specifies the default units to be assumed when the design is analyzed. After setting a unit with this command, all subsequent analysis will operate based on the units provided.

Note: This command operates silently and does not return direct feedback of its operation.

Arguments

-capacitance *value* - (Optional) Specify the unit of capacitance in Farads. Valid values range from kilofarads (kF) to femtofarads (fF). The default unit of capacitance is picofarads (pF).

-time *value* - (Optional) Specify the unit of time in seconds. Valid values range from kilosecond (ks) to femtosecond (fs). The default unit of time is nanosecond (ns).

-current *value* - (Optional) Specify the default unit of current in amperes. Valid values range from kiloAmps (kA) to femtoAmps (fA). The default unit of amperes is milliAmps (mA).

-voltage *value* - (Optional) Specify the default unit of voltage in Volts. Valid values range from kilovolts (kV) to femotovolts (fV). The default unit of voltage is Volts (V).

-power *value* - (Optional) Specify the default unit of power in watts. Valid values range from kilowatts (kW) to femtowatts (fW). The default unit of power is milliwatts (mW).

-resistance *value* - (Optional) Specify the default unit of resistance in ohms. Valid values range from kilo-ohm (kOhm) to femto-ohm (fOhm). The default unit of resistance is ohms (Ohm).

-suffix *value* - (Optional) Specify the suffix to be used for the specified units.

-digits *value* - (Optional, Default Value of 1) Specify the number of digits to be used when printing units.

-quiet - (Optional) This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

Specify that voltage should be in millivolts and all values should use three digits

```
set_units -voltage mV -digits 3
```

After issuing the command above, subsequently change the default unit for current to Amperes

```
set_units -current A
```

The second time that set_units is issued above does not over-ride or change the values set in the first instance

See Also

- set_operating_conditions
- set_property
- set_hierarchy_separator

split_diff_pair_ports

Remove differential pair relationship between 2 ports

Syntax

```
split_diff_pair_ports [-quiet] ports ...
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
ports	no		Ports to split

Categories

PinPlanning

start_gui

Start planAhead GUI

Syntax

```
start_gui
```

Returns

Nothing

Categories

GUIControl

Description

This command launches the PlanAhead tool GUI mode when the tool is running in batch mode. The GUI is invoked with the current project, design, and run information.

Example

The following example is executed from the command prompt when the PlanAhead tool is running in batch mode:

```
PlanAhead% start_gui
```

See Also

stop_gui

startgroup

Start a set of commands that can be undone/redone as a group

Syntax

```
startgroup [-try] [-quiet]
```

Returns

int

Usage

Name	Optional	Default	Description
-try	yes		Don't start a group if one has already been started
-quiet	yes		Ignore command errors

Categories

GUIControl

Description

This command will define a sequence of commands to undo or redo as a series. Use this command with endgroup to create the sequence of commands.

Note: While the PlanAhead tool can support multiple groups of commands to **undo** or **redo**, it cannot support nested command groups. You must use **endgroup** to end a command sequence prior to using **startgroup** to create a new command sequence

The **startgroup** command returns an integer value of 0 if a group is already started, and returns an integer value of 1 if the startgroup command has started a new group.

Arguments

-try - Returns 1 if a new group is started. Returns 0 if a group has already been started, and therefore a new group cannot be started.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example defines a startgroup, executes a sequence of related commands, and then executes the endgroup. This sequence of commands can be undone or redone as a group:

```
startgroup
create_pblock pblock_wbArbEngine
create_pblock pblock_usbEngnSRM
add_cells_to_pblock pblock_wbArbEngine [get_cells [list wbArbEngine]] -clear_locs
add_cells_to_pblock pblock_usbEngnSRM [get_cells [list usbEngine1/usbEngineSRAM]] \
-clear_locs
endgroup
```

See Also

- endgroup
- redo
- startgroup

stop_gui

Close planAhead GUI

Syntax

```
stop_gui
```

Returns

Nothing

Categories

GUIControl

Description

This command stops the PlanAhead tool GUI mode and places the tool into a batch mode of operation. In batch mode, all commands must be entered as Tcl commands or through Tcl scripts.

Example

The following example stops and closes the PlanAhead tool GUI and places the tool into batch mode:

```
stop_gui
```

See Also

start_gui

swap_locs

Swap two locations

Syntax

```
swap_locs [-quiet] aloc bloc
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
aloc	no		First location (port/cell/site - should be of same type as 'bloc')
bloc	no		Second location (port/cell/site - should be of same type as 'aloc')

Categories

Floorplan

Description

Swap the LOC constraints assigned to two similar logic elements. A logic element is an element that can be placed onto a device resource on the FPGA.

Some DRC checking is performed when the **swap_locs** command is executed to ensure that the two selected elements can in fact be assigned to their new locations. If the location of either element is invalid for any reason, the **swap_locs** command will fail and an error will be returned.

Arguments

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

aloc - The location of the first logic element to swap. This can be specified as a port, a cell, or a device site.

bloc - The location of the second logic element to swap. This can be specified as a port, a cell, or a device site. This must match the type specified by the *aloc* variable.

Example

The following example swaps the instances assigned to the two specified device sites:

```
swap_locs [get_sites {OLOGIC_X2Y1}] [get_sites {OLOGIC_X2Y0}]
```

See Also

- get_cells
- get_ports
- get_sites

undo

Un-do previous command

Syntax

```
undo [-list] [-quiet]
```

Returns

with -list, the list of undoable tasks

Usage

Name	Optional	Default	Description
-list	yes		Show a list of undoable tasks
-quiet	yes		Ignore command errors

Categories

GUIControl

Description

This command will undo a prior command execution. This command can be used repeatedly to undo a series of commands.

If a group of commands has been created using the **startgroup** and **endgroup** commands, this command will undo that group as a sequence. The undo command will start at the **endgroup** command and continue to undo until it hits the **startgroup** command.

If you **undo** a command, and then change your mind, you can **redo** the command.

Arguments

-list - Reports the list of commands that can be undone. As you execute the undo command, the PlanAhead tool will step backward through this list of commands.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example returns a list of commands that can be undone:

```
undo -list
```

See Also

- endgroup
- redo
- startgroup

unhighlight_objects

Unhighlight objects that are currently highlighted

Syntax

```
unhighlight_objects [-color_index arg] [-rgb args] [-color arg]
[-quiet] [objects]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-color_index	yes		Color index
-rgb	yes		RGB color index list
-color	yes		Name of color
-quiet	yes		Ignore command errors
objects	yes		Objects to unhighlight

Categories

GUIControl

Description

This command is for use in the PlanAhead tool GUI mode. This command unhighlights the specified object or objects that were previously highlighted by the **highlight_objects** command.

This command supports the color options as specified below. However, these options are not necessary to unhighlight a specific object, but can be used to unhighlight all objects currently highlighted in the specified color. See the example below.

Arguments

-color_index arg - Specify the color index to unhighlight. The color index is defined by the Highlight category of the Tools > Options > Themes command. Refer to the *PlanAhead User Guide* (ug632) for more information on setting themes.

-rgb args - Specify the color to unhighlight in the form of an RGB code specified as {R G B}. For instance, {255 255 0} specifies the color yellow.

-color arg - Specify the color to unhighlight. Supported highlight colors are red, green, blue, magenta, yellow, cyan, and orange.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

objects - Specifies one or more objects to be unhighlighted. If no objects are specified, all highlighted objects of the specified color will be unhighlighted. If no color is specified, all highlighted objects will be unhighlighted.

Examples

The following example unhighlights the selected objects:

```
unhighlight_objects [get_selected_objects]
```

The following example unhighlights all objects currently highlighted in the color yellow:

```
unhighlight_objects -color yellow
```

See Also

- get_selected_objects
- highlight_objects

unmark_objects

Unmark items that are currently marked

Syntax

```
unmark_objects [-rgb args] [-color arg] [-quiet] [objects]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-rgb	yes		RGB color index list
-color	yes		Name of color
-quiet	yes		Ignore command errors
objects	yes		Objects to unmark

Categories

GUIControl

Description

This command is for use in the PlanAhead tool GUI mode. This command unmarks the specified object or objects that were previously marked by the **mark_objects** command.

This command supports the color options as specified below. However, these options are not necessary to unmark a specific object, but can be used to unmark all objects currently marked in the specified color. See the example below.

Arguments

-rgb args - Specify the color to unmark in the form of an RGB code specified as {R G B}. For instance, {255 255 0} specifies the color yellow.

-color arg - Specify the color to unmark. Supported highlight colors are red, green, blue, magenta, yellow, cyan, and orange.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

objects - Specifies one or more objects to be unmarked. If no objects are specified, all marked objects of the specified color will be unmarked. If no color is specified, all marked objects will be unmarked.

Examples

The following example unmarks the selected objects:

```
unmark_objects [get_selected_objects]
```

The following example unmarks all objects currently marked in the color yellow:

```
unmark_objects -color yellow
```

See Also

- get_selected_objects
- mark_objects

unselect_objects

Unselect items that are currently selected

Syntax

```
unselect_objects [-quiet] [objects]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
objects	yes		Objects to unselect

Categories

GUIControl

Description

This command unselects the specified object or objects that were previously selected by the **select_objects** command.

The **unselect_objects** command will unselect both primary and secondary selected objects. The selection of secondary objects is controlled through the use of Selection Rules defined in the **Tools > Options** command. Refer to the *PlanAhead User Guide* (ug632) for more information on Setting Selection Rules.

Arguments

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

objects - Specifies one or more objects to be unselected. If no objects are specified, all selected objects will be unselected.

Examples

The following example unselects the specified site on the device:

```
unselect_objects [get_sites SLICE_X56Y214]
```

The following example unselects all currently selected objects:

```
unselect_objects
```

See Also

- get_selected_objects
- select_objects

update_file

Update an imported file with the contents of a remote file

Syntax

```
update_file -file arg -remote_file arg [-quiet]
```

Returns

the file that was updated

Usage

Name	Optional	Default	Description
-file	no		Imported file to update
-remote_file	no		Remote file to import
-quiet	yes		Ignore command errors

Categories

Project

Description

Update a single file with the contents of a specified remote file. Note that this command can be used to update a local file with the contents of its original remote file, or replace it with the contents of a different remote file.

Arguments

-file arg - Specifies the local project file to be updated.

-remote_file arg - Specifies the path and name of a remote file to replace the local file with. The remote file is copied into the local project directory structure and added to the project, replacing the specified file.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Examples

The following example updates the specified file with the remote file:

```
update_file -file C:/Data/design1.v -remote_file C:/Source/design1.v
```

Note: No warnings will be issued for newer local files that will be overwritten.

See Also

reimport_files

upgrade_ip

Update a configurable IP to a later version

Syntax

```
upgrade_ip [-srcset arg] -files args [-quiet]
```

Returns

list of files that were updated

Usage

Name	Optional	Default	Description
-srcset	yes		Source set name
-files	no		IP source files to be updated
-quiet	yes		Ignore command errors

Categories

Project

verify_config

Analyze implemented runs to ensure they follow rules required for partial reconfiguration

Syntax

```
verify_config [-file arg] [-verbose] [-quiet] runs ...
```

Returns

pr report

Usage

Name	Optional	Default	Description
-file	yes		Output report file name
-verbose	yes		Output verbose information in log file
-quiet	yes		Ignore command errors
runs	no		List of implemented Runs to be verified

Categories

PartialReconfiguration

version

Returns the build for planAhead and the build date

Syntax

```
version [-quiet]
```

Returns

planAhead version

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors

Categories

Report

wait_on_run

Block execution of further Tcl commands until the specified run completes.

Syntax

```
wait_on_run [-timeout arg] [-quiet] run
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-timeout	yes	-1	Maximum time to wait for the run to complete (in minutes)
-quiet	yes		Ignore command errors
run	no		Run to wait on

Categories

Project

Description

Block execution of Tcl commands until the specified run completes, or until the specified amount of time has elapsed.

Note: This command is used for running the PlanAhead tool in batch mode or from Tcl scripts. It is ignored when running interactively from the GUI.

Arguments

-timeout *arg* - Specifies the time in minutes that the **wait_on_run** command should wait until the run finishes. This allows you to define a period of time beyond which the PlanAhead tool should resume executing Tcl commands even if the specified run has not finished execution. The default value of -1 is used if timeout is not specified, meaning that there is no limit to the amount of time the PlanAhead tool will wait for the run to complete.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

run - Specifies the name of the run that the PlanAhead tool should wait on. The specified run must be actively running when the **wait_on_run** command is used, or the PlanAhead tool will simply return an error.

Example

The following example launches the impl_1 run, and instructs the PlanAhead tool to wait for the specified run to complete, or to wait for one hour, whichever occurs first:

```
launch_runs impl_1
wait_on_run -timeout 60 impl_1
```

See Also

launch_runs

write_bitstream

Write a bitstream for the current design

Syntax

```
write_bitstream [-bitgen_options arg] [-quiet] [file]
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-bitgen_options	yes		Command line options for bitgen
-quiet	yes		Ignore command errors
file	yes		The name of the .bit file to generate

Categories

Project, FileIO

Description

Writes a bitstream file for the current project by calling Bitgen from within the PlanAhead tool. This command must be run on an Implemented Design. The bitstream written will be based on the open Implemented Design.

Arguments

-bitgen_options *arg* - specify one or more command line options for the BitGen command. See the *Command Line Tools Users Guide* (ug628) for more information on valid arguments.

Note: The PlanAhead tool uses the -intstyle pa option when calling BitGen. This option should not be specified under bitgen_options or an error will occur.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

file - Specifies the name of the bitstream file to write. The default extension for a bitstream file is .bit, however you must specify the extension for the file when using the write_bitstream command.

Note: If the path is not specified as part of the file name, the PlanAhead tool will write the named file into the current working directory, or the directory from which the PlanAhead tool was launched.

Examples

The following example writes a bitstream file of the specified name:

```
write_bitstream design1.bit
```

The following example writes a bitstream file to the specified folder, and specifies the -d and -l command-line options for BitGen:

```
write_bitstream -bitgen_options {-d -l -g compress -g crc:disable} C:/Data/design.bit
```

Note: The -d command-line option indicates that DRC should not be run prior to writing the bitstream file, and the -l option creates an ASCII logic allocation file (*file*.ll) for the current design. Also note that the two configuration (-g) options, Compress and CRC, are specified with a separate -g argument for each option.

See Also

- launch_runs
- open_impl_design

write_chipscope_cdc

Export nets that are connected to debug ports

Syntax

```
write_chipscope_cdc [-quiet] file
```

Returns

name of the output file

Usage

Name	Optional	Default	Description
-quiet	yes		Ignore command errors
file	no		ChipScope CDC file name

Categories

FileIO, ChipScope

Description

Causes the PlanAhead tool to write a ChipScope Definition and Connection (CDC) file containing the debug cores, ports, and signals defined in the current project.

ChipScope debug cores are added to a project through the use of the create_debug_core command. The CDC file stores information about source files, destination files, core parameters, and core settings for the ChipScope Pro Analyzer.

You can import this CDC file into the ChipScope Analyzer to automatically set up the net names on the ILA core data and trigger ports. The written CDC file can also be used as input to other projects in the PlanAhead tool with the use of the read_chipscope_cdc command.

Arguments

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

file - Specifies the file name to be written.

Note: If the path is not specified as part of the file name, the PlanAhead tool will write the named file into the current working directory, or the directory from which the PlanAhead tool was launched.

Example

The following example writes a CDC file called bft.cdc:

```
write_chipscope_cdc bft.cdc
```

The written CDC file will include signals to be debugged by Chipscope as well as the clock domain for the signals, and other settings appropriate for use in Chipscope.

See Also

- create_debug_core
- read_chipscope_cdc

write_csv

Export package pin and port placement information

Syntax

```
write_csv [-mode arg] [-force] [-quiet] file
```

Returns

name of the output file

Usage

Name	Optional	Default	Description
-mode	yes	port	Output Mode
-force	yes		Overwrite existing file
-quiet	yes		Ignore command errors
file	no		Pin Planning CSV file

Categories

FileIO

Description

Export package pin and port placement information into a comma separated value (CSV) file.

The specific format and requirements of the CSV file are described in the *PlanAhead Users Guide* (ug632.pdf).

Arguments

-force - Forces the overwrite of an existing CSV file of the same name.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

file - Specifies the file name of the CSV file to be exported.

Note: If the path is not specified as part of the file name, the PlanAhead tool will write the named file into the current working directory, or the directory from which the PlanAhead tool was launched.

Example

The following example exports a CSV file from the current project:

```
write_csv C:/Data/pinList.csv
```

See Also

read_csv

write_edif

Export the current netlist as an EDIF file

Syntax

```
write_edif [-pblocks args] [-cell arg] [-force] [-quiet] file
```

Returns

the name of the output file or directory

Usage

Name	Optional	Default	Description
-pblocks	yes		Export netlist for these pblocks (not valid with -cell)
-cell	yes		Export netlist for this cell (not valid with -pblocks)
-force	yes		Overwrite existing file
-quiet	yes		Ignore command errors
file	no		Output file (directory with -pblocks or -cell)

Categories

FileIO

Description

Export the current netlist as an EDIF file, or output the contents of specific Pblocks or hierarchical cells as EDIF netlist files.

Arguments

-pblocks *arg* - Instructs the PlanAhead tool to output the contents of the specified Pblocks as EDIF netlist files. The contents of each Pblock will be written to a separate EDIF file.

-cell *arg* - Instructs the PlanAhead tool to output the contents of the specified hierarchical cell as EDIF netlist files. Only one cell can be specified for output.

Note: The -pblock and -cell arguments are mutually exclusive. Although they are optional arguments, only one may be specified at one time.

-force - Forces the overwrite of an existing EDIF file or files of the same name.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

file - Specifies the name of the EDIF file to write. The default file extension for an EDIF netlist is .edn.

Note: If the path is not specified as part of the file name, the PlanAhead tool will write the named file into the current working directory, or the directory from which the PlanAhead tool was launched.

In the case of either the -pblock or -cell argument being used, this argument specifies a directory name where the EDIF netlist files for each Pblock or cell will be written. The EDIF netlist file will be named after the Pblock or cell. If the directory specified does not exist, the PlanAhead tool will return an error.

Examples

The following example writes an EDIF netlist file for the whole design to the specified file name:

```
write_edif C:/Data/edifOut.edn
```

The following example outputs an EDIF netlist for all Pblocks in the design. The files will be written to the specified directory.

```
write_edif -pblocks [get_pblocks] C:/Data/FPGA_Design/
```

write_ibis

Write IBIS models for current floorplan

Syntax

```
write_ibis [-allmodels] [-nopin] [-truncate arg] [-ibs arg]
[-pkg arg] [-quiet] file
```

Returns

name of the output file

Usage

Name	Optional	Default	Description
-allmodels	yes		Include all available buffer models for this architecture. By default, only buffer models used by the floorplan are included.

Categories

FileIO

Description

Export the IBIS models for the target device in the current design. the PlanAhead tool combines the netlist and implementation details from the design with the per-pin parasitic package information to create a custom IBIS model for the design.

Because the write_ibis command incorporates design information into the IBIS Model, you must have an RTL, Netlist, or Implemented Design open when running this command.

Arguments

-allmodels - Export all buffer models for the target device. By default the PlanAhead tool will only write buffer models used by the design.

-nopin - Disable per-pin modeling of the path from the die pad to the package pin. The IBIS model will include a single RLC transmission line model representation for all pins in the [Package] section. By default the PlanAhead tool includes per-pin modeling of the package as RLC matrices in the [Define Package Model] section if this data is available.

-truncate *arg* - Specifies the maximum length for a signal name in the output file. Names longer than this will be truncated. Legal values are 20, 40, or 0 (unlimited). By default the PlanAhead tool will truncate signal names to 40 characters in accordance with the IBIS version 4.2 specification.

-ibs *arg* - Specify an updated generic IBIS models file. This is used to override the IBIS models found in the the PlanAhead tool installation under the parts directory. This argument is required for any parts that do not have generic models in the installation directory.

-pkg *arg* - Specify an updated per pin parasitic package data file. This is used to override the parasitic package file found in the the PlanAhead tool installation hierarchy under the parts directory. This argument is required for any parts that do not have generic models in the installation directory.

file - Specifies the file name of the IBIS file to be exported.

Note: If the path is not specified as part of the file name, the PlanAhead tool will write the named file into the current working directory, or the directory from which the PlanAhead tool was launched.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example exports all buffer models for the target device, eliminates truncation of signal names, and specifies the file name and path to write:

```
write_ibis -allmodels -truncate 0 C:/Data/FPGA_Design/ibisOut.txt
```

write_ncd

Export the placement to a ncd file

Syntax

```
write_ncd [-force] [-quiet] file
```

Returns

name of the output file

Usage

Name	Optional	Default	Description
-force	yes		Overwrite existing file
-quiet	yes		Ignore command errors
file	no		Output file

Categories

FileIO

Description

This command exports a Xilinx Native Circuit Description (NCD) file from an implemented design. The NCD file is created by MAP and PAR during implementation, and converts the logical Netlist Design into a physical design implementing the targeted Xilinx device architecture.

Arguments

-force - Forces the overwrite of an existing NCD file of the same name.

file - Specifies the file name of the NCD file to be exported.

Note: If the path is not specified as part of the file name, the PlanAhead tool will write the named file into the current working directory, or the directory from which the PlanAhead tool was launched.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example exports an NCD file from the current implemented design:

```
write_ncd C:/Data/FPGA_Design/designOut.ncd
```

write_pcf

Export the transformed constraints to a physical constraints file (pcf)

Syntax

```
write_pcf [-force] [-quiet] file
```

Returns

name of the output file

Usage

Name	Optional	Default	Description
-force	yes		Overwrite existing file
-quiet	yes		Ignore command errors
file	no		Output file

Categories

FileIO

Description

This command creates a PCF file from the current constraints in the design. A PCF file is an ASCII file with physical constraints defined by MAP, followed by physical constraints defined by the user. The MAP section is rewritten during every implementation pass. The order of constraints ensures that user constraints are read last, and will override MAP constraints.

The PCF file is an optional input to PAR, the FPGA Editor, TRACE, NetGen, and BitGen.

Arguments

-force - Forces the overwrite of an existing PCF file of the same name.

file - Specifies the file name of the PCF file to be exported.

Note: If the path is not specified as part of the file name, the PlanAhead tool will write the named file into the current working directory, or the directory from which the PlanAhead tool was launched.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

Example

The following example creates a PCF file called `designOut.pcf`:

```
write_pcf designOut.pcf
```

See Also

- read_ucf
- write_ucf

write_sdf

write_sdf command generates flat sdf delay files for event simulation

Syntax

```
write_sdf [-process_corner arg] [-cell arg]
[-rename_top_module arg] [-quiet] file
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-process_corner	yes	slow	Specify process corner for which SDF delays are required; Values: slow, fast
-cell	yes	whole design	Root of the design to write. eg. des.subblk.cpu
-rename_top_module	yes	new top module name	Replace name of top module with custom name e.g. netlist
-quiet	yes		Ignore command errors
file	no		file name

Categories

FileIO, Simulation

Description

Write the timing delays for cells in the design to a Standard Delay Format (SDF) file.

The output SDF file can be used by the **write_verilog** command to create Verilog netlists for static timing analysis and timing simulation.

Arguments

-process_corner [fast | slow] - Write delays for a specified process corner. Delays are greater in the slow process corner than in the fast process corner. Valid values are 'slow' or 'fast'. By default, the SDF file is written for the slow process corner.

-cell *arg* - Write the SDF file from a specific cell of the design hierarchy. The default is to create an SDF file for the whole design.

-rename_top_module *arg* - Change the name of the top module in the output SDF file.

-force - Forces the overwrite of an existing SDF file of the same name.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

file - Specifies the file name of the SDF file to be exported. The SDF file is referenced in the Verilog netlist by the use of the **-sdf_anno** and **-sdf_file** arguments of the **write_verilog** command.

Note: If the path is not specified as part of the file name, the PlanAhead tool will write the named file into the current working directory, or the directory from which the PlanAhead tool was launched.

Example

The following example writes an SDF file to the specified directory:

```
write_sdf C:/Data/FPGA_Design/designOut.sdf
```

See Also

write_verilog

write_timing

Export a set of timing results to file

Syntax

```
write_timing [-force] [-quiet] name file
```

Returns

Nothing

Usage

Name	Optional	Default	Description
-force	yes		Overwrite existing file
-quiet	yes		Ignore command errors
name	no		Name for the set of results
file	no		Name of the file to write the results to

Categories

FileIO

Description

Write the results of timing analysis to the specified file. This command writes the results from timing analysis previously created by the **report_timing** command; it does not actually run timing analysis.

write_timing produces a legacy timing path report. The format of this file is different from the output of the **report_timing -file** command.

Arguments

-force - Forces the overwrite of an existing file of the same name.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

name - Specify the name of the results set created by the **report_timing** command. These are the timing results that will be output to the specified file.

file - Specifies the file name of the timing file to be written.

Note: If the path is not specified as part of the file name, the PlanAhead tool will write the named file into the current working directory, or the directory from which the PlanAhead tool was launched.

Example

The following example runs timing analysis and creates a timing results set called `time_1`, then writes the named timing results to the specified file:

```
report_timing -name time_1
write_timing time_1 C:/Data/FPGA_Design/bft_time_1.txt
```

See Also

report_timing

write_ucf

Export UCF information to a file or directory

Syntax

```
write_ucf [-no_fixed_only] [-constraints arg] [-pblocks args]
[-cell arg] [-quiet] file
```

Returns

name of the output file or directory

Usage

Name	Optional	Default	Description
-no_fixed_only	yes		Export fixed and non-fixed placement (by default only fixed placement will be exported)
-constraints	yes	valid	Include constraints that are flagged invalid Values: valid, invalid, all
-pblocks	yes		Export placement for these pblocks (not valid with -cell)
-cell	yes		Export placement for this cell (not valid with -pblocks)
-quiet	yes		Ignore command errors
file	no		Output file (directory with -pblocks, -cell)

Categories

FileIO

Description

Export physical constraints to a user constraint file (UCF). The UCF can be exported from the top-level, which is the default, from specific Pblocks, or from a specific hierarchical cell.

This command will export the constraints from all UCF files of the active constraint fileset. Constraints from multiple files will be included in the specified UCF file.

Arguments

-no_fixed_only - Export both fixed and unfixed placement LOCs to the constraint file being written. By default only the fixed LOCs will be written to the UCF file. Fixed LOCs are associated with user-assigned placements, while unfixed LOCs are associated with tool assigned placements.

-constraints *arg* - Export constraints that are flagged valid, invalid, or all constraints (both valid and invalid). The default behavior is to export only valid constraints to the UCF file. However, you can specify -constraints values of VALID, INVALID, or ALL.

-pblocks *arg* - Specify one or more Pblocks to export the constraints from.

-cell *arg* - Specify the name of a hierarchical cell in the current design to export the constraints from. The constraints will be written to the specified UCF file relative to the specified cell.

Note: A design must be open when using the -cell option.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

file - Specifies the name of the UCF file to be created.

Note: If the path is not specified as part of the file name, the PlanAhead tool will write the named file into the current working directory, or the directory from which the PlanAhead tool was launched.

In the case of either the -pblock or -cell argument being used, this argument specifies a directory name where the UCF files for each Pblock or the cell will be written. The UCF file will be named after the Pblock or cell. If the directory specified does not exist, the PlanAhead tool will return an error.

Example

The following example writes the valid and invalid constraints, including both fixed and unfixed LOCs, for all Pblocks found in the design to the specified directory:

```
write_ucf -no_fixed_only -constraints all -pblocks [get_pblocks] C:/Data/FPGA_Design
```

See Also

- read_ucf
- save_design
- save_design_as

write_verilog

Export the current netlist in Verilog format

Syntax

```
write_verilog [-cell arg] [-mode arg] [-lib]
[-write_all_overrides] [-rename_top_module arg] [-sdf_anno arg]
[-sdf_file arg] [-force] [-quiet] file
```

Returns

the name of the output file or directory

Usage

Name	Optional	Default	Description
-cell	yes	whole design	Root of the design to write. eg. des.subblk.cp u
-mode	yes	design	Values: design, port, sta, sim
-lib	yes		Write each library into a separate file
-write_all_overrides	yes		Write parameter overrides on Xilinx primitives even if the override value is the same as the default value
-rename_top_module	yes	new top module name	Replace top module name with custom name e.g. netlist
-sdf_anno	yes		Specify if sdf_annotate system task statement is generated
-sdf_file	yes	file .sdf	Full path to sdf file location
-force	yes		Overwrite existing file
-quiet	yes		Ignore command errors
file	no		Which file to write

Categories

FileIO, Simulation

Description

Write a Verilog netlist of the current design or from a specific cell of the design to the specified file or directory. The output is a IEEE 1364-2001 compliant Verilog HDL file that contains netlist information obtained from the input design files.

You can output a complete netlist of the design or specific cell, or output a port list for the design, or a Verilog netlist for simulation or static timing analysis.

Arguments

-cell *arg* - Write the Verilog netlist from a specified cell or block level of the design hierarchy. The output Verilog file or files will only include information contained within the specified cell or module.

-mode *arg* - Specifies the mode to use when writing the Verilog file. By default, the Verilog netlist is written for the whole design. Legal mode values are:

- design - Output a verilog netlist for the whole design. This acts as a snapshot of the design, including all post placement, implementation, and routing information in the netlist.
- port - Outputs only the I/O ports for the top-level of the design.
- sta - Output a Verilog netlist to be used for static timing analysis (STA).
- sim - Output the Verilog netlist to be used as a simulation model. The output netlist is not suitable for synthesis.

-security_mode *arg* - Valid values are **all**, **secured_only**, or **unsecured_only**. The default mode is all, and writes both encrypted and unencrypted cells to the specified output file. If any cells require encryption, the whole design is written to an encrypted zip file. When **secured_only** is specified, only the portions of the design that must be encrypted are written to a zip file. When **unsecured_only** is specified only the portions of the design that do not require encryption are written to an unencrypted text file.

-lib - Causes a separate Verilog file to be output for each library used by the design.

Note: This option is the opposite of, and replaces the **-nolib** option from prior releases. Previously the default behavior of **write_verilog** was to output a separate Verilog file for each library used in the design, unless **-nolib** was specified. Now you must specify the **-lib** option to output separate Verilog files for each library

-write_all_overrides [true | false] - This option accepts a value of **true** or **false**. If you specify **true** then every parameter override in the design is written to the Verilog output even if the value of the parameter is the same as the defined primitive default value. If the option is **false** then parameter values which are equivalent to the primitive defaults are not output to the Verilog file. Setting this option to **true** will not change the result but makes the output Verilog more verbose.

-rename_top *arg* - This option only works with **-mode sim** to allow the Verilog netlist to plug into top-level simulation testbenches. The name of the top module output to the Verilog file is changed to the specified top module name.

-sdf_anno [true | false] - This option only works with **-mode sim**, and is set to false by default. This option specifies that the **$sdf_annotate** statement should be added to the Verilog netlist when true. Legal values are true (or 1) and false (or 0).

-sdf_file *arg* - This option specifies the path and file name of the SDF file to use when writing the **$sdf_annotate** statement into the ouptut Verilog file. When not specified, the SDF file is assumed to have the same name and path as the Verilog output specified by <*file*>, with a file extension of `.sdf`. The SDF file must be separately written to the specified file path and name using the **write_sdf** command.

-force - Forces the overwrite of an existing Verilog file or files of the same name.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

file - Specifies the file name of the Verilog file to be exported. If the file name does not have either a `.v` or `.ver` file extension then the name is assumed to refer to a directory, and the Verilog file is named after the top module, and is output to the specified directory.

Note: If the path is not specified as part of the file name, the PlanAhead tool will write the named file into the current working directory, or the directory from which the PlanAhead tool was launched.

Examples

The following example writes a Verilog simulation netlist file for the whole design to the specified file and path:

```
write_verilog C:/Data/my_verilog.v
```

In the following example, the specified file does not have a `.v` or `.ver` file extension, and so is treated as a directory name. A Verilog file is created in the specified directory and is named after the top module in the design. In addition, because the **-mode sim** and **-sdf_anno** options are specified, the **$sdf_annotate** statement will be added to the Verilog netlist. However, since the **-sdf_file** option is not specified, the SDF file is assumed to have the same name as the Verilog output file, with an `.sdf` file extension:

```
write_verilog C:/Data/my_verilog.net -mode sim -sdf_anno true
```

Note: The Verilog file name in this example will be `top.v` and will be written to the `my_verilog.net` directory. The SDF file written to the **$sdf_annotate** statement is assumed to be in the same directory and named `top.sdf`

See Also

- write_sdf
- write_vhdl

write_vhdl

Export the current netlist in VHDL format

Syntax

```
write_vhdl [-cell arg] [-mode arg] [-lib]
[-write_all_overrides] [-arch_only] [-force] [-quiet] file
```

Returns

the name of the output file or directory

Usage

Name	Optional	Default	Description
-cell	yes	whole design	Root of the design to write. eg. des.subblk.cpu
-mode	yes	sim	Values: sim, port
-lib	yes		Write each library into a separate file
-write_all_overrides	yes		Write parameter overrides on Xilinx primitives even if the same as the default value
-arch_only	yes		Write only the architecture, not the entity declaration for the top cell
-force	yes		Overwrite existing file
-quiet	yes		Ignore command errors
file	no		Which file to write

Categories

FileIO, Simulation

Description

Write a VHDL netlist of the current design or from a specific cell of the design to the specified file or directory.

The output of this command is a VHDL IEEE 1076.4 VITAL-2000 compliant VHDL file that contains netlist information obtained from the input design files. You can output a complete netlist of the design or specific cell, or output a port list for the design.

Arguments

-cell *arg* - Write the VHDL netlist from a specified cell or block level of the design hierarchy. The output VHDL file or files will only include information contained within the specified cell or module.

-mode *arg* - Specifies the mode to use when writing the VHDL file. By default, the simulation netlist is written for the whole design. Legal mode values are:

- sim - Output the VHDL netlist to be used as a simulation model. The output netlist is not suitable for synthesis. This is the default setting.
- port - Outputs only the I/O ports in the entity declaration for the top-level module.

-security_mode *arg* - Valid values are **all**, **secured_only**, or **unsecured_only**. The default mode is all, and writes both encrypted and unencrypted cells to the specified output file. If any cells require encryption, the whole design is written to an encrypted zip file. When **secured_only** is specified, only the portions of the design that must be encrypted are written to a zip file. When **unsecured_only** is specified only the portions of the design that do not require encryption are written to an unencrypted text file.

-lib - Causes a separate VHDL file to be output for each library used by the design.

Note: This option is the opposite of, and replaces the **-nolib** option from prior releases. Previously the default behavior of **write_vhdl** was to output a separate VHDL file for each library used in the design, unless **-nolib** was specified. Now you must specify the **-lib** option to output separate files for each library

-write_all_overrides [true | false] - This option accepts a value of **true** or **false**. If you specify **true** then every parameter override in the design is written to the VHDL output even if the value of the parameter is the same as the defined primitive default value. If the option is **false** then parameter values which are equivalent to the primitive defaults are not output to the VHDL file. Setting this option to **true** will not change the result but makes the output netlist more verbose.

-rename_top *arg* - This option only works with **-mode sim** to allow the VHDL netlist to plug into top-level simulation testbenches. The name of the top module output to the VHDL file is changed to the specified top module name.

-arch_only - Suppresses the entity definition of the top-level module, and outputs the architecture only. This simplifies the use of the output VHDL netlist with a separate testbench.

-force - Forces the overwrite of an existing VHDL file or files of the same name.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

file - Specifies the file name of the VHDL file to be exported. If the file name does not have either a `.vhd` or `.vhdl` file extension then the name is assumed to be a directory, and the VHDL file is named after the top module, and is output to the specified directory.

Note: If the path is not specified as part of the file name, the PlanAhead tool will write the named file into the current working directory, or the directory from which the PlanAhead tool was launched.

Examples

The following example writes a VHDL simulation netlist file for the whole design to the specified file and path:

```
write_vhdl C:/Data/bft_top.vhd
```

In the following example the entity definition of the top-level module is not output to the VHDL netlist:

```
write_vhdl C:/Data/vhdl_arch_only.vhd -arch_only
```

See Also

write_verilog

write_xdc

Specifies the name of the XDC file to be created. The default file extension for a XDC file is .xdc. If you called this command with -pblocks or -cell The file argument will specify the output directory

Syntax

```
write_xdc [-no_fixed_only] [-constraints arg] [-pblocks args]
[-cell arg] [-sdc] [-no_tool_comments] [-force] [-quiet] file
```

Returns

name of the output file or directory

Usage

Name	Optional	Default	Description
-no_fixed_only	yes		Export fixed and non-fixed placement (by default only fixed placement will be exported)
-constraints	yes	valid	Include constraints that are flagged invalid Values: valid, invalid all
-pblocks	yes		Export placement for these pblocks (not valid with -cell)
-cell	yes		Export placement for this cell (not valid with -pblocks)
-sdc	yes		Export all timing constraints
-no_tool_comments	yes		Don't write verbose tool generated comments to the xdc when translating from ucf
-force	yes		Overwrite existing file
-quiet	yes		Ignore command errors
file	no		Output file/directory path. (path is a directory if -pblocks or -cell used)

Categories

FileIO

Description

Export physical constraints to a Xilinx Design Constraints file (XDC). The XDC can be exported from the top-level, which is the default, from specific Pblocks, or from a specific hierarchical cell.

This command can be used to create an XDC file from a design with UCF files. All constraints from the active constraint fileset will be exported to the XDC, even if they come from multiple files.

Arguments

-no_fixed_only - Export both fixed and unfixed placement LOCs to the constraint file being written. By default only the fixed LOCs will be written to the XDC file. Fixed LOCs are associated with user-assigned placements, while unfixed LOCs are associated with tool assigned placements.

-constraints *arg* - Export constraints that are flagged valid, invalid, or all constraints (both valid and invalid). The default behavior is to export only valid constraints to the XDC file. However, you can specify -constraints values of VALID, INVALID, or ALL.

-pblocks *arg* - Specify one or more Pblocks to export the constraints from.

-cell *arg* - Specify the name of a hierarchical cell in the current design to export the constraints from. The constraints will be written to the specified XDC file relative to the specified cell.

Note: A design must be open when using the -cell option.

-sdc - Exports only the timing constraints in an SDC format from the current design. Does not export any other defined constraints.

-no_tool_comments - Disable generation of tool comments when writing the XDC file. Without this argument the PlanAhead tool will add comments and warnings related to the creation of the XDC file.

-force - Forces the overwrite of an existing XDC file or files of the same name.

-quiet - This option tells the PlanAhead tool to execute the command quietly, ignore any command line errors, and return no error messages if the command fails to execute.

file - Specifies the name of the XDC file to be created. The default file extension for a XDC file is .ucf.

Note: If the path is not specified as part of the file name, the PlanAhead tool will write the named file into the current working directory, or the directory from which the PlanAhead tool was launched.

In the case of either the -pblock or -cell argument being used, this argument specifies a directory name where the XDC files for each Pblock or the cell will be written. The XDC file will be named after the Pblock or cell. If the directory specified does not exist, the PlanAhead tool will return an error.

Example

The following example writes the valid and invalid constraints, including both fixed and unfixed LOCs, for all Pblocks found in the design to the specified directory:

```
write_xdc -no_fixed_only -constraints all -pblocks [get_pblocks] C:/Data/FPGA_Design
```

XILINX®

Appendix

Additional Resources

The *PlanAhead User Guide (UG632)* provides detailed information on usage of the PlanAhead tool. The *Hierarchical Design Methodology Guide (UG748)* provides information on working with hierarchical designs for Xilinx FPGA devices.

Other Xilinx resources on the Web include:

- **Xilinx Glossary -** http://www.xilinx.com/company/terms.htm
- **Xilinx Support and Documentation -** http://www.xilinx.com/support

Tcl Developer Xchange

Tcl reference material is available on the Internet. Xilinx recommends the Tcl Developer Xchange, which maintains the open source code base for Tcl, and is located at:

http://www.tcl.tk

An introductory tutorial is available at:

http://www.tcl.tk/man/tcl/tutorial/tcltutorial.html

About SDC

Synopsys Design Constraints (SDC) is an accepted industry standard for communicating design intent to tools, particularly for timing analysis. A reference copy of the SDC specification is available from Synopsys by registering for the TAP-in program at:

http://www.synopsys.com/Community/Interoperability/Pages/TapinSDC.aspx