PlanAhead Tcl Command Reference Guide

XILINX®

Notice of Disclaimer

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at http://www.xilinx.com/warranty.htm; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: http://www.xilinx.com/warranty.htm#critapps.

© Copyright 2002-2013 Xilinx Inc. All rights reserved. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. The PowerPC name and logo are registered trademarks of IBM Corp., and used under license. All other trademarks are the property of their respective owners.

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 application programming interfaces, and is used by 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 within the PlanAhead tool.

Note This manual is not a comprehensive reference for the Tcl language. It is a reference to the specific capabilities of the Tcl shell within the PlanAhead tool, and provides reference to additional Tcl programming resources.

Launching the PlanAhead tool

You can launch the PlanAhead tool and access the features of the tool using different methods depending on your preference.

Tcl Shell Mode

If you prefer to work directly with Tcl commands, you can interact with your design using Tcl commands with one of the following methods:

• Enter individual Tcl commands in the PlanAhead tool Tcl shell outside of the GUI.

• Enter individual Tcl commands in the Tcl Console at the bottom of the GUI.

• Run Tcl scripts from the PlanAhead tool Tcl shell.

• Run Tcl scripts from the PlanAhead GUI.

Use the following command to invoke the PlanAhead tool Tcl shell either at the Linux command prompt or within a Windows Command Prompt window:

```
planahead –mode tcl
```

Tip On Windows, you can also select **Start > All Programs > Xilinx Design Tools > ISE Design Suite 14.5 > Accessories > ISE Design Suite 64/32 Bit Command Prompt**.

Tcl Batch Mode

You can use the PlanAhead tool in batch mode by supplying a Tcl script when invoking the tool. Use the following command either at the Linux command prompt or within a Windows Command Prompt window:

```
planahead –mode batch –source <your_Tcl_script>
```

The PlanAhead tool Tcl shell will open, run the specified Tcl script, and exit when the script completes. In batch mode, you can queue up a series of Tcl scripts to process a number of designs overnight through synthesis, simulation, and implementation, and review the results on the following morning.

PlanAhead GUI Mode

If you prefer to work in a GUI, you can launch the PlanAhead tool from Windows or Linuxin GUI mode.

Launch the PlanAhead tool from your working directory. By default the PlanAhead journal and log files, and any generated report files, are written to the directory from which the tool is launched. This makes it easier to locate the project file, log files, and journal files, which are written to the launch directory.

In the Windows OS, you can also select **Start > All Programs > Xilinx Design Tools > ISE Design Suite 14.5 > PlanAhead > PlanAhead**.

Tip You can also double-click the PlanAhead shortcut icon on your Windows desktop.

In the Linux OS, enter the following command at the command prompt:

```
planahead
-or-
planahead –mode gui
```

If you need help, with the PlanAhead tool command line executable, type:

```
planahead -help
```

If you are running the PlanAhead tool from a Tcl shell, you can open the PlanAhead GUI directly from the Tcl shell by using the **start_gui** command.

From the PlanAhead GUI, you can close the GUI and return to a PlanAhead Tcl shell by using the **stop_gui** command.

Tcl Journal Files

When you invoke the PlanAhead tool, it writes the `planahead.log` file to record the various commands 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. The journal file can be used as a source to create new Tcl scripts.

Note Backup versions of the journal file, named `planahead_<id>.backup.jou`, are written to save the details of prior runs whenever the PlanAhead tool is launched. The *<id>* is a unique identifier that allow the tool to create and store multiple backup versions of the log and journal 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 -args *command* — Provides an abbreviated help text for the specified command, including the command syntax and a brief description of each argument.

  help -args get_cells

- help -syntax *command* — Reports the command syntax for the specified command.

  help -syntax get_cells

Tcl Initialization Scripts

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

1. In the software installation:
   installdir/*version*/ISE_DS/PlanAhead/scripts/init.tcl

2. In the local user directory:
   - For Windows 7: %APPDATA%/Roaming/Xilinx/PlanAhead/init.tcl
   - For Windows XP: %APPDATA%/Xilinx/PlanAhead/init.tcl
   - For Linux: $HOME/.Xilinx/PlanAhead/init.tcl

Where:

installdir is the installation directory where the PlanAhead tool is installed.

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 local user 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 the enterprise `init.tcl` script.

- The `init.tcl` file in the home directory allows each user to specify additional commands, or to override commands from the software installation to meet their specific design requirements.

- No `init.tcl` file is provided with the PlanAhead tool installation. You must create the `init.tcl` file and place it in either the installation directory, or your home directory, as discussd to meet your specific needs.

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

Note You can also specify the **-init** option when launching the PlanAhead tool from the command line. Type **planahead -help** for more information.

Sourcing a Tcl Script

A Tcl script can be sourced from either one of the command-line options or from the GUI. Within the PlanAhead GUI you can source a Tcl script from **Tools > Run Tcl Script**.

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

```
source file_name
```

When you invoke a Tcl script from the PlanAhead GUI, a progress bar is displayed and all operations in the IDE are blocked until the scripts completes.

There is no way to interrupt script execution during run time; consequently, standard OS methods of killing a process must be used to force interruption of the tool. If the process is killed, you lose any work done since your last save.

Typing **help source** in the Tcl console will provide additional information regarding the **source** command.

General Tcl Syntax Guidelines

Tcl uses the Linux file separator (/) convention regardless of which Operating System you are running.

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

Sourcing a Tcl Script

A Tcl script can be sourced from either one of the command-line options or from the GUI. Within the PlanAhead GUI you can source a Tcl script from **Tools > Run Tcl Script**.

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

```
source file_name
```

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.

There is no way to interrupt script execution during run time; consequently, standard OS methods of killing a process must be used to force interruption of the tool. If the process is killed, you lose any work done since your last save.

Typing **help source** in the Tcl console will provide additional information regarding the **source** command.

Using Tcl Eval

When executing Tcl commands, you can use variable substitution to replace some of the command line arguments accepted or required by the Tcl command. However, you must use the Tcl **eval** command to evaluate the command line with the Tcl variable as part of the command.

For instance, the help command can take the **-category** argument, with one of a number of command categories as options:

```
help -category ipflow
```

You can define a variable to hold the command category:

```
set cat "ipflow"
```

You can then evaluate the variable in the context of the Tcl command:

```
eval help -category $cat
```

or,

```
set cat "-category ipflow"
eval help $cat
```

You can also use braces {} in place of quotation marks "" to achieve the same result:

```
set runblocksOptDesignOpts { -sweep -retarget -propconst -remap }
eval opt_design $runblocksOptDesignOpts
```

Typing **help eval** in the Tcl console will provide additional information regarding the **eval** command.

General Syntax Structure

The general structure of the PlanAhead tool 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.

First Class Tcl Objects and Relationships

The Tcl commands in the PlanAhead tool 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 tool; 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 groups 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 list of wires that eventually be physically connected directly together. Nets can be hierarchical or flat, but always sorts a list 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 lists 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.
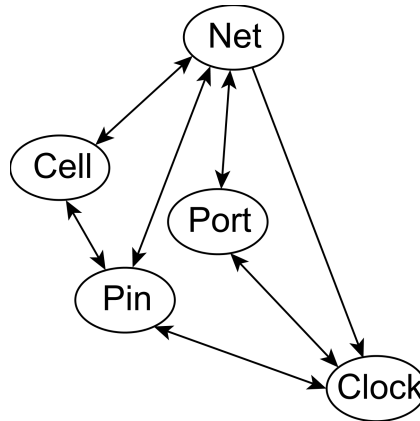
Errors, Warnings, Critical Warnings, and Info Messages

Messages that result from individual commands appear in the `planahead.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", or "ERROR" followed by a subsystem identifier and a unique number.

The following example shows an INFO message that appears after inferring timing constraints:

```
[Timing 38-33] Done inferring timing constraints.
```

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.

Chapter 2

Tcl Commands Listed by Category

Categories

- ChipScope
- DRC
- FileIO
- Floorplan
- GUIControl
- IPFlow
- IPIntegrator
- Netlist
- Object
- PartialReconfiguration
- Partition
- PinPlanning
- Power
- Project
- PropertyAndParameter
- Report
- SDC
- Simulation
- SysGen
- Timing
- ToolLaunch
- Tools
- XDC
- XPS

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
- write_debug_probes

DRC

- create_drc_check
- create_drc_violation
- delete_drc_check
- report_drc
- reset_drc
- reset_drc_check

FileIO

- infer_diff_pairs
- read_chipscope_cdc
- read_csv
- read_edif
- read_ip
- read_pxml
- read_twx
- read_ucf
- read_verilog
- read_vhdl
- read_xdl
- write_chipscope_cdc
- write_csv
- write_debug_probes
- write_edif
- write_ibis
- 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
- unplace_cell

GUIControl

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

IPFlow

- copy_ip
- create_ip
- generate_target
- get_ipdefs
- get_ips
- import_ip
- open_example_project
- read_ip
- reset_target
- update_ip_catalog
- upgrade_ip

IPIntegrator

generate_target

Netlist

- create_pin
- remove_pin
- rename_ref
- resize_net_bus
- resize_pin_bus

Object

- create_drc_check
- delete_drc_check
- filter
- get_bel_pins
- get_bels
- get_boards
- get_cells
- get_clock_regions
- get_clocks
- get_debug_cores
- get_debug_ports
- get_delays
- get_designs
- get_files
- get_filesets
- get_interfaces
- get_io_standards
- get_iobanks
- get_ipdefs
- get_ips
- get_lib_cells
- get_lib_pins
- get_libs
- get_nets
- get_nodes
- get_package_pins
- get_parts
- get_path_groups
- get_pblocks
- get_pins
- get_pips
- get_ports
- get_projects
- get_property
- get_reconfig_modules
- get_runs
- get_selected_objects
- get_site_pins
- get_site_pips
- get_sites
- get_slrs
- get_tiles

- get_wires
- list_property
- list_property_value
- report_property
- reset_drc_check
- reset_property
- set_property

PartialReconfiguration

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

Partition

- config_partition
- demote_run
- promote_run

PinPlanning

- create_interface
- create_port
- delete_interface
- make_diff_pair_ports
- place_ports
- remove_port
- resize_port_bus
- set_package_pin_val
- split_diff_pair_ports

Power

- delete_power_results
- report_power
- reset_default_switching_activity
- reset_operating_conditions
- reset_switching_activity
- set_operating_conditions
- set_power_opt
- set_switching_activity

Project

- add_files
- archive_project
- close_design
- close_project
- copy_ip
- create_fileset
- create_project
- create_run
- current_fileset
- current_project
- current_run
- delete_fileset
- delete_run
- find_top
- generate_target
- get_boards
- get_files
- get_filesets
- get_ips
- get_projects
- get_runs
- help
- import_as_run
- import_files
- import_ip
- import_synplify
- import_xise
- import_xst
- launch_runs
- list_targets
- make_wrapper

- open_example_project
- open_io_design
- open_project
- open_rtl_design
- open_run
- refresh_design
- reimport_files
- remove_files
- reorder_files
- reset_project
- reset_run
- reset_target
- save_constraints
- save_constraints_as
- save_project_as
- set_speed_grade
- update_compile_order
- update_files
- wait_on_run

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

- create_drc_violation
- create_slack_histogram
- delete_timing_results
- delete_utilization_results
- get_msg_count
- get_msg_limit

- report_carry_chains
- report_clock_interaction
- report_clocks
- report_config_timing
- report_control_sets
- report_datasheet
- report_debug_core
- report_default_switching_activity
- report_drc
- report_environment
- report_high_fanout_nets
- report_io
- report_operating_conditions
- report_param
- report_power
- report_property
- report_pulse_width
- report_resources
- report_ssn
- report_sso
- report_stats
- report_switching_activity
- report_timing
- report_transformed_primitives
- report_utilization
- reset_drc
- reset_msg_count
- reset_msg_limit
- reset_msg_severity
- reset_ssn
- reset_sso
- reset_timing
- set_msg_limit
- set_msg_severity
- version

SDC

- all_clocks
- all_fanin
- all_fanout
- all_inputs
- all_outputs
- all_registers
- current_design
- current_instance
- get_cells
- get_clocks
- get_hierarchy_separator
- get_nets
- get_pins
- get_ports
- report_operating_conditions
- reset_operating_conditions
- set_hierarchy_separator
- set_logic_unconnected
- set_operating_conditions

Simulation

- add_files
- create_fileset
- delete_fileset
- import_files
- launch_isim
- launch_modelsim
- remove_files
- report_simlib_info
- reset_simulation
- write_verilog
- write_vhdl

SysGen

- create_sysgen
- make_wrapper

Timing

- config_timing_analysis
- config_timing_corners
- delete_timing_results
- report_config_timing
- report_timing
- reset_timing
- set_delay_model
- update_timing

ToolLaunch

- crossprobe_fed
- launch_chipscope_analyzer
- launch_fpga_editor
- launch_impact
- launch_isim
- launch_modelsim
- launch_sdk
- launch_xpa

Tools

link_design

XDC

- add_cells_to_pblock
- all_clocks
- all_cpus
- all_dsps
- all_fanin
- all_fanout
- all_ffs
- all_hsios
- all_inputs
- all_latches
- all_outputs
- all_rams
- all_registers
- create_pblock
- current_design
- current_instance
- delete_pblock
- filter
- get_cells
- get_clocks
- get_hierarchy_separator
- get_iobanks
- get_nets
- get_package_pins
- get_path_groups
- get_pblocks
- get_pins
- get_ports
- get_sites
- remove_cells_from_pblock
- resize_pblock
- set_external_delay
- set_hierarchy_separator
- set_logic_unconnected
- set_operating_conditions
- set_package_pin_val
- set_power_opt
- set_switching_activity

XPS

- create_xps
- export_hardware
- generate_target
- get_boards
- launch_sdk
- list_targets
- make_wrapper
- reset_target

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 [**-top**] [**-add_primitives**] [**-clear_locs**]
[**-quiet**] [**-verbose**] *pblock* [*cells...*]

Returns

Nothing

Usage

Name	Description
[**-top**]	Add the top level instance; This option can't be used with -cells, or -add_primitives options. You must specify either -cells or -top option.
[**-add_primitives**]	Assign all the primitives of the specified instances to a pblock
[**-clear_locs**]	Clear instance location constraints
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
pblock	Pblock to add cells to
[*cells*]	Cells to add. You can't use this option with -top option. You must specify either -cells or -top option.

Categories

Floorplan, XDC

Description

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 **resize_pblock** command. The **resize_pblock** command can also be used to manually move and resize pblocks.

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

Arguments

-top - (Optional) Add the top level instance to create a Pblock for the whole design. You must either specify *cells* or the **-top** option to add objects to the Pblock.

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

Note This option cannot be used with **-top**.

-clear_locs - (Optional) 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

pblock - The name assigned to the Pblock.

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

Note If -top is specified, you cannot also specify *cells*

Examples

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] [-verbose] [files...]
```

Returns

List of file objects that were added

Usage

Name	Description
[**-fileset**]	Fileset name
[**-norecurse**]	Do not recursively search in specified directories
[**-scan_for_includes**]	Scan and add any included files found in the fileset's RTL sources
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*files*]	Name of the files and/or directories to add. Must be specified if -scan_for_includes is not used.

Categories

Project, Simulation

Description

Adds one or more source files, or the source file contents of one or more directories, to the specified fileset in the current project.

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* - (Optional) The fileset to which the specified source files should be added. An error is returned if the specified fileset does not exist. If no fileset is specified the files are added to the source fileset by default.

-norecurse - (Optional) Do not recurse through subdirectories of any specified directories. Without this argument, the tool searches through any subdirectories for additional source files that can be added to a project.

-scan_for_includes - (Optional) Scan Verilog source files for any **'include** statements and add these referenced files to the specified fileset. By default, **'include** files are not added to the fileset.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

files - One or more file names or directory names to be added to the fileset. If a directory name is specified, all valid source files found in the directory, and in subdirectories of the directory, are added to the fileset.

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

Examples

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

```
add_files rtl.v
```

In the preceding example the tool looks for the rtl.v file in the current working directory since no file path is specified, and the file is 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 addition, the tool ignores 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.

The following example adds an existing IP core file (XCO) to the current project:

```
add_files -norecurse C:/Data/ip/c_addsub_v11_0_0.xco
```

Note Use the **import_ip** command to import the IP file into the local project folders

The following example adds an existing Embedded Processor sub-design into the current project:

```
add_files C:/Data/dvi_tpg_demo_ORG/system.xmp
```

Note Use the **create_xps** command to create a new Embedded Processor using Xilinx Platform Studio (XPS)

The following example adds an existing DSP module, created in System Generator, into the current project:

```
add_files C:/Data/model1.mdl
```

Note Use the **create_sysgen** command to use System Generator to create a new DSP module

See Also

- create_sysgen
- create_xps
- import_files
- import_ip
- read_ip
- read_verilog
- read_vhdl

all_clocks

Get a list of all clocks in the current design.

Syntax

all_clocks [**-quiet**] [**-verbose**]

Returns

List of clock objects

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

SDC, XDC

Description

Returns a list of all 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

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 applies the **set_propagated_clock** command to all clocks, and also demonstrates how the returned list (**all_clocks**) can be passed to another command:

```
% set_propagated_clock [all_clocks]
```

See Also

get_clocks

all_cpus

Get a list of cpu cells in the current design.

Syntax

all_cpus [**-quiet**] [**-verbose**]

Returns

List of cpu cell objects

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

XDC

Description

Returns 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.

The **all_cpus** command is scoped to return the objects hierarchically, from the top-level of the design or from the level of the current instance. By default the current instance is defined as the top level of the design, but can be changed by using the **current_instance** command.

Note This command returns a list of CPU cell objects

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

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_dsps
- all_hsios
- all_registers
- current_instance
- get_cells

all_dsps

Get a list of dsp cells in the current design.

Syntax

all_dsps [**-quiet**] [**-verbose**]

Returns

List of dsp cell objects

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

XDC

Description

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

The **all_dsps** command is scoped to return the objects hierarchically, from the top-level of the design or from the level of the current instance. By default the current instance is defined as the top level of the design, but can be changed by using the **current_instance** command.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

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
- current_instance
- get_cells

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]
[-verbose] to
```

Returns

List of cell or pin objects

Usage

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

Categories

SDC, XDC

Description

Returns a list of port, pin or cell objects in the fan-in of the specified sinks.

The **all_fanin** command is scoped to return objects from current level of the hierarchy of the design, either from the top-level or from the level of the current instance. By default the current instance is defined as the top level of the design, but can be changed by using the **current_instance** command. To return the fan-in across all levels of the hierarchy, use the **-flat** option.

Arguments

-startpoints_only - (Optional) Find only the timing start points. When this option is used, none of the intermediate points in the fan-in network are returned. This option can be used to identify the primary driver(s) of the sinks.

-flat - (Optional) Ignore the hierarchy of the design. By default, only the objects at the same level of hierarchy as the sinks are returned. When using this option, all the objects in the fan-in network of the sinks are considered, regardless of hierarchy.

-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) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

to - The pins, ports, or nets from which you want the fan-in objects reported.

Examples

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

```
all_fanin [get_ports led_pins[*] ]
```

The following example traces back from the clock pin of the specified flip- flop to the clock source (an MMCM output pin in this example):

```
all_fanin -flat -startpoints_only [get_pins cmd_parse_i0/prescale_reg[7]/C]
```

See Also

- all_fanout
- current_instance
- get_cells
- get_pins
- get_ports

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**] [**-verbose**] *from*

Returns

List of cell or pin objects

Usage

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

Categories

SDC, XDC

Description

Returns a list of port, pin, or cell objects in the fanout of the specified sources.

The **all_fanout** command is scoped to return objects from current level of the hierarchy of the design, either from the top-level or from the level of the current instance. By default the current instance is defined as the top level of the design, but can be changed by using the **current_instance** command. To return the fanout across all levels of the hierarchy, use the **-flat** option.

Arguments

-endpoints_only - (Optional) Find only the timing endpoints. When this option is used, none of the intermediate points in the fan-out network are returned. This option can be used to identify the primary loads of the drivers.

-flat - (Optional) Ignore the hierarchy of the design. By default, only the objects at the same level of hierarchy as the sinks are returned. When using this option, all the objects in the fan-out network of the drivers are considered, regardless of hierarchy.

-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) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

from - The source ports, pins, or nets from which to list the objects in the fanout path.

Examples

The following example gets the fanout for all input ports in the design:

```
all_fanout [all_inputs]
```

See Also

- all_fanin
- current_instance
- get_cells
- get_pins
- get_ports

all_ffs

Get a list of flip flop cells in the current design.

Syntax

all_ffs [**-quiet**] [**-verbose**]

Returns

List of flip flop cell objects

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

XDC

Description

Returns a list of all flip flop instances in the current design.

The **all_ffs** command is scoped to return the objects hierarchically, from the top-level of the design or from the level of the current instance. By default the current instance is defined as the top level of the design, but can be changed by using the **current_instance** command.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example returns the count of all flops in the design, then returns the count of all flops in the fftEngine module:

```
current_instance
INFO: [PlanAhead 12-618] Current instance is the top level of design 'netlist_1'.
top
llength [all_ffs]
15741
current_instance fftEngine
fftEngine
llength [all_ffs]
1519
```

The following example reports the currently assigned properties on the specified flop:

```
report_property [lindex [all_ffs] 2 ]
```

See Also

- all_latches
- all_registers
- current_instance
- get_cells
- report_property

all_hsios

Get a list of hsio cells in the current design.

Syntax

all_hsios [**-quiet**] [**-verbose**]

Returns

List of hsio cell objects

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

[XDC](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.

The **all_hsios** command is scoped to return the objects hierarchically, from the top-level of the design or from the level of the current instance. By default the current instance is defined as the top level of the design, but can be changed by using the **current_instance** command.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

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

```
all_hsios
```

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

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

See Also

- all_cpus
- all_dsps
- all_registers
- get_cells

all_inputs

Get a list of all input ports in the current design.

Syntax

all_inputs [**-quiet**] [**-verbose**]

Returns

List of port objects

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

SDC, XDC

Description

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

The **all_inputs** command is scoped to return the objects hierarchically, from the top-level of the design or from the level of the current instance. By default the current instance is defined as the top level of the design, but can be changed by using the **current_instance** command.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example returns all input port objects 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_clocks
- all_outputs
- current_instance
- get_clocks
- get_ports

all_latches

Get a list of all latch cells in the current design.

Syntax

all_latches [**-quiet**] [**-verbose**]

Returns

List of latch cell objects

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

XDC

Description

Returns a list of all latches that have been declared in the current design.

The **all_latches** command is scoped to return the objects hierarchically, from the top-level of the design or from the level of the current instance. By default the current instance is defined as the top level of the design, but can be changed by using the **current_instance** command.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

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

```
all_latches
```

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

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

See Also

- all_ffs
- all_registers
- current_instance
- get_cells

all_outputs

Get a list of all output ports in the current design.

Syntax

all_outputs [**-quiet**] [**-verbose**]

Returns

List of port objects

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

SDC, XDC

Description

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

The **all_outputs** command is scoped to return the objects hierarchically, from the top-level of the design or from the level of the current instance. By default the current instance is defined as the top level of the design, but can be changed by using the **current_instance** command.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

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
- current_instance
- get_ports

all_rams

Get a list of ram cells in the current design.

Syntax

all_rams [**-quiet**] [**-verbose**]

Returns

List of ram cell objects

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

[XDC](#)

Description

Returns a list of all the RAM cell objects present in the current instance, including Block RAMS, Block RAM FIFOs, and Distributed RAMS. These RAM cell objects can be assigned to a variable or passed into another command.

The **all_rams** command is scoped to return the objects hierarchically, from the top-level of the design or from the level of the current instance. By default the current instance is defined as the top level of the design, but can be changed by using the **current_instance** command.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example returns all RAM objects in the design:

```
all_rams
```

The following example sets the current instance, and returns all RAM objects hierarchically from the level of the current instance:

```
current_instance usbEngine0
all_rams
```

See Also

- all_clocks
- all_cpus
- all_dsps
- all_fanin
- all_fanout
- all_ffs
- all_hsios
- all_inputs
- all_latches
- all_outputs
- all_registers
- current_instance
- get_cells

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]
[-no_hierarchy] [-quiet] [-verbose]
```

Returns

List of cell or pin objects

Usage

Name	Description
[**-clock**]	Consider registers of this clock
[**-rise_clock**]	Consider registers triggered by clock rising edge
[**-fall_clock**]	Consider registers triggered by clock falling edge
[**-cells**]	Return list of cells (default)
[**-data_pins**]	Return list of register data pins
[**-clock_pins**]	Return list of register clock pins
[**-async_pins**]	Return list of async preset/clear pins
[**-output_pins**]	Return list of register output pins
[**-level_sensitive**]	Only consider level-sensitive latches
[**-edge_triggered**]	Only consider edge-triggered flip-flops
[**-no_hierarchy**]	Only search the current instance
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

SDC, XDC

Description

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

The list of returned objects can be limited by the use of the 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 get a list of the pins of collected registers instead of the register objects by specifying one or more of the pin arguments.

Arguments

-cells - (Optional) Return 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) Return a list of all registers whose clock pins are in the fanout of the specified clock.

-rise_clock *args* - (Optional) Return a list of registers triggered by the rising edge of the specified clocks.

-fall_clock *args* - (Optional) Return a list of registers triggered by the falling edge of the specified clocks.

Note Do not combine **-clock**, **-rise_clock**, and **-fall_clock** in the same command.

-level_sensitive - (Optional) Return a list of the level-sensitive registers or latches.

-edge_triggered - (Optional) Return a list of the edge-triggered registers or flip-flops.

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

-clock_pins - (Optional) Return a list of clock pins of the registers that meet the search requirement.

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

-output_pins - (Optional) Return a list of output pins of the registers that meet the search requirement.

Note Use the ***_pins** arguments separately. If you specify multiple arguments, only one argument is applied in the following order of precedence: **-data_pins**, **-clock_pins**, **-async_pins**, **-output_pins**.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

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_msg_limit

archive_project

Archive the current project.

Syntax

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

Returns

True

Usage

Name	Description
[**-force**]	Overwrite existing archived file
[**-exclude_run_results**]	Exclude run results from the archive
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*file*]	Name of the archive file

Categories

Project

Description

Archives a project to store as backup, or to encapsulate the design and send it to a remote site. The 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 - (Optional) Overwrite an existing ZIP file of the same name. If the ZIP file exists, the tool returns an error unless the **-force** argument is specified.

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

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

file - (Optional) 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 is created.

Examples

The following command archives the current project:

```
archive_project
```

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

See Also

current_project

close_design

Close the current design.

Syntax

```
close_design [-quiet] [-verbose]
```

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Project

Description

Closes the currently active design. 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** to save any changes made to the design before using the **close_design** command.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example closes the current design:

```
close_design
```

Note If multiple designs are open, you can specify the current design 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
```

current_design sets **rtl_1** as the active design, then the **close_design** command closes it.

See Also

current_design

close_project

Close current opened project.

Syntax

close_project [**-delete**] [**-quiet**] [**-verbose**]

Returns

Nothing

Usage

Name	Description
[**-delete**]	Delete the project from disk also
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Project

Description

Closes the current open project.

Arguments

-delete - Delete the project data from the hard disk after closing the project.

Note Use this argument with caution. You will not be prompted to confirm the deletion of project data.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following command closes the active project:

```
close_project
```

This example closes the current 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 current project, and then closes the project and deletes it from the computer hard disk:

```
current_project project_1
close_project -delete
```

Note Use the **-delete** argument with caution. You will not be prompted to confirm the deletion of project data.

See Also

current_project

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]
[-verbose] run
```

Returns

Nothing

Usage

Name	Description
[-cell]	Partition instance to configure in the given run. In order to modify top Partition do not specify this option.
[-reconfig_module]	Reconfigurable Module variant to apply to this instance in this run
[-import]	Set this instance (or static logic) to 'import' action for this run
[-implement]	Set this instance (or static logic) to 'implement' action for this run
[-import_dir]	Directory from which to import this previously implemented module
[-preservation]	Set the preservation level for the Partition. Values: routing, placement, synthesis Default: routing
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
run	Run to be modified

Categories

[PartialReconfiguration](), [Partition]()

Description

Configure a partition prior to synthesis or implementation.

Specify the partition module as a cell or as the top module, the action to take for the specific module, the path to import partition data from, and the level of partition data to preserve.

Arguments

-cell - (Optional) Specify the cell name of the partition to configure.

Note If this option is not specified, the **config_partition** command applies to the top partition in the design.

-reconfig_module *name* - (Optional) The Reconfigurable Module variant to configure.

-import - (Optional) Import data for the specified partition from the **-import_dir** during synthesis or implementation to preserve prior results.

-implement -(Optional) Discard prior results and reimplement the specified partition during synthesis or implementation. This treats the partition like any other hierarchical module in the design.

Note **-implement** and **-import** are mutually exclusive arguments.

-import_dir *name* - (Optional) The directory to import partition data from.

Note If the path to the directory is not specified, the tool will look in your home directory for the partition data:

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

-preservation *level* - (Optional) The preservation level for the imported partitions. Valid values are **routing**, **placement**, and **synthesis**. The default is to preserve routing.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

run - (Required) The run to be configured.

Examples

The following example creates a partition on the module sub_block_inst and configures the partition to be implemented during impl_1 run:

```
config_partition -run synth_1 -cell usbEngine0 -import -import_dir \
C:/Data/DP_RTL/synth1 -preservation placement
```

See Also

promote_run

config_timing_analysis

Configure timing analysis general settings.

Syntax

```
config_timing_analysis [-enable_input_delay_default_clock arg]
[-enable_preset_clear_arcs arg] [-disable_flight_delays arg]
[-quiet] [-verbose]
```

Returns

Nothing

Usage

Name	Description
[**-enable_input_delay_default _clock**]	Launch SDC unclocked input delays from an internally defined clock: Values: true, false; This option is not supported for UCF constraints
[**-enable_preset_clear_arcs**]	Time paths through asynchronous preset or clear timing arcs: true, false;
[**-disable_flight_delays**]	Disable adding package times to IO Calculations : Values: true, false;
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Timing

Description

This command configures general features of timing analysis.

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

Arguments

-enable_input_delay_default_clock [true | false] - (Optional) Launch unclocked input delays from an internally defined clock for timing analysis. The valid values are true or false, with a default setting of false.

Note This argument is not supported in the PlanAhead tool

-enable_preset_clear_arcs [true | false] - (Optional) Time paths through asynchronous preset or clear timing arcs. The valid values are true or false, with a default setting of false.

-disable_flight_delays [true | false] - (Optional) Do not add package delays to I/O calculations when this option is true.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example ignores unrelated clocks in UCF designs:

```
config_timing_analysis -disable_paths_between_unrelated_ucf_clocks true
```

See Also

- config_timing_corners
- report_timing

config_timing_corners

Configure single / multi corner timing analysis settings.

Syntax

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

Returns

Nothing

Usage

Name	Description
[**-corner**]	Name of the timing corner to be modified : Values: Slow, Fast
[**-delay_type**]	Type of path delays to be analysed for specified timing corner: Values: none, max, min, min_max
[**-setup**]	Enable timing corner for setup analysis (equivalent to -delay_type max)
[**-hold**]	Enable timing corner for hold analysis (equivalent to -delay_type min)
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Timing

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

Note The names of the corners are case sensitive.

-delay_type *value* - (Optional) Specify the type of path delays to be analyzed 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) Execute the command quietly, returning no messages from the
command. The command also returns TCL_OK regardless of any errors encountered
during execution.

Note Any errors encountered on the command-line, while launching the command,
will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages
from this command.

Note Message limits can be defined with the **set_msg_limit** command.

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
- report_timing

connect_debug_port

Connect nets and pins to debug port channels.

Syntax

connect_debug_port [**-channel_start_index** *arg*] [**-quiet**]
[**-verbose**] *port nets...*

Returns

Nothing

Usage

Name	Description
[**-channel_start_index**]	Connect nets starting at channel index
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
port	Debug port name
nets	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. ChipScope debug cores can be inserted into a design with the **create_debug_core** or **read_chipscope_cdc** commands.

If you try to connect too many signals to a port, or there are not enough channels to support the connection, the 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* - 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 tool will place connections on the first available channel index.

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

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

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

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_i m0_we_i ]] -channel_start_index 2
```

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

See Also

- create_debug_core
- create_debug_port
- disconnect_debug_port
- get_debug_ports
- get_nets
- implement_debug_core
- read_chipscope_cdc
- set_property

copy_ip

Copy an existing IP.

Syntax

```
copy_ip -name arg [-quiet] [-verbose] objects...
```

Returns

IP file object that was added to the project

Usage

Name	Description
-name	Name of copied IP
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
objects	IP to be copied

Categories

Project, IPFlow

Description

Create a copy of an IP core that has been previously instanced into the current project.

Arguments

-name *arg* - (Required) Specify the name of the new IP to be created.

-dir *arg* - (Optional) The path to a directory outside of the local project to store the newly created IP. The specified directory must already exist, or the command will return an error.

Note If a directory is not specified, the new IP will be added to the local project directory structure at *project_name*.srcs/sources_1/ip.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

object - (Required) The IP object to copy. The **copy_ip** command can only be used to copy a single IP core at one time. The IP must be specified as an IP object returned by the get_ips command, and not simply referenced by name.

Example

The following example create a copy of the FIFO core previously instanced into the current project and writes it to the specified directory:

```
copy_ip -name newFIFO -dir C:/Data/new_IP [get_ips oldFIFO]
```

See Also

- create_ip
- get_ips
- import_ip
- read_ip

create_debug_core

Create a new ChipScope debug core.

Syntax

create_debug_core [**-quiet**] [**-verbose**] *name type*

Returns

New debug_core object

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
name	Name of the new debug core instance
type	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.

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

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.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

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

type - The ChipScope debug core to insert. Currently the chipscope_ila_v1 core is supported in the 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 tool also adds an Integrated Controller core (chipscope_icon_v1) as a container for one or more ILA cores. However, you cannot directly add an ICON core to the project.

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
```

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**] [**-verbose**] *name type*

Returns

New debug_port object

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
name	Name of the debug core instance
type	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

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

name - 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 - 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.

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_drc_check

Create a user defined drc rule.

Syntax

```
create_drc_check [-category arg] -name arg [-desc arg]
[-msg arg] [-rule_body arg] [-quiet] [-verbose]
```

Returns

Nothing

Usage

Name	Description
[**-category**]	Specify the category for this rule. This is the major grouping for your rule. It is optional and will default to User Defined. Default: User Defined
-name	Specify the name for this rule. This must be of the form PREFIX-id where XXXX is a 4-6 letter abbreviation and id is an integer identifying a particular rule. Similar rules should have the same abbreviation and each a unique id.
[**-desc**]	Specify the short description for this rule. It is optional and will default to . Default: User rule - default description
[**-msg**]	Specify the full description for this rule. Including the substitutions. Values are: %MSG_STRING %NETLIST_ELEMENT %SITE_GROUP %CLOCK_REGION %BANK.
[**-rule_body**]	The string representing the body of the rule. This can be a tcl proc name or any string of tcl code to be evaluated.
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

DRC, Object

Description

Create a new user-defined DRC rule check, drc_check, for use by the tool when running **report_drc**.

This command allows you to define a unique name or abbreviation for the user-defined rule check, optionally group the rule into a special category and provide a description of the rule, define a general placeholder message for the check when violations are encountered, and refer to the Tcl code associated with the design rule check to be run during the **report_drc** command.

The general placeholder message defined in this command is populated with specific information related to the design objects and violations found by the Tcl checker procedure, and by the **create_drc_violation** command.

The process in brief is:

- Write a Tcl checker procedure to define the method applied when checking the user-defined rule, and the objects to check against the rule. The Tcl checker procedure is defined in a separate Tcl script that must be loaded by the **source** command prior to running **report_drc**.

- Use **create_drc_violation** in the Tcl checker to identify and flag violations found when checking the rule against a design.

- Define a user-defined DRC rule check using the **create_drc_check** command that calls the Tcl checker proc from the **-rule_body**.

- Create a rule deck using the **create_drc_ruledeck** command, and add the user-defined rule check to the rule deck using the **add_drc_checks** command.

- Run **report_drc**, and specify either the rule deck, or the user-defined rule check to check for violations.

If a drc_check of the specified name is already defined in the tool, an error is returned. In this case, to overwrite or redefine and existing drc_check, you must first delete the check using the delete_drc_check command.

The DRC rule check object features the **is_enabled** property that can be set to TRUE or FALSE using the **set_property** command. When a new rule check is created, the **is_enabled** property is set to TRUE as a default. Set the **is_enabled** property to FALSE to disable the rule check from being used when **report_drc** is run. This lets you create new DRC checks, add them to rule decks using add_drc_checks, and then enable them or disable them as needed without having to remove them from the rule deck.

Arguments

-category *arg* - (Optional) Defines a grouping for the rule. The default is "User Defined". This is used as the first level of hierarchy in the GUI when listing DRC rules.

-name *arg* - The unique name for the design rule. This should match the name used by the **create_drc_violation** commands in the Tcl checker procedure specified in **-rule_body**. The name will appear in the DRC report with any associated violations. The name should consist of a short 4 to 6 letter abbreviation for the rule group, and an ID to differentiate it from other checks in the same group, for instance ABCD-1 or ABCD-23.

-desc *arg* - (Optional) A brief description of the rule. The default is "User Rule". This is displayed when listing DRC rules in the GUI. The description is also used in the DRC report and summary.

-msg *arg* - (Optional) This is the message displayed when a violation of the rule is found. The message can include placeholders for dynamic substitution with design elements found in violation of the rule. The design data is substituted into the message at the time **report_drc** is run. Each substitution key has a long form, and a short form as shown below. Valid substitutions keys are:

- %MSG_STRING (%STR) - This is the message string defined by the **-msg** option in the **create_drc_violation** command for the specific violation.

  Note %STR is the default message for the **create_drc_check** command if the **-msg** option is not specified. In this case, any message defined by create_drc_violation in the **-rule_body** is simply passed through to the DRC report.

- %NETLIST_ELEMENT (%ELG) - Netlist elements including cells, pins, ports, and nets.

- %SITE_GROUP (%SIG) - Device site.

- %CLOCK_REGION (%CRG) - Clock region.

- %BANK (%PBG) - Package IO bank.

-rule_body *arg* - (Optional) This is the name of the Tcl procedure which defines the rule checking functionality. The Tcl procedure can be embedded here, into the **-rule_body** option, or can be separately defined in a Tcl script that must be loaded with the **source** command when the tool is launched, or prior to running the **report_drc** command.

The Tcl checker procedure can create DRC violation objects, using the create_drc_violation command, containing the design elements that are associated with a design rule violation. The tool populates the substitution keys in the message defined by **-msg** with the design elements from the violation object.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example defines a new design rule check named RAMW-1, with the category and description defined, using the default severity of Warning, and calling the dataWidthCheck procedure when the check is run:

```
create_drc_check -name {RAMW-1} -category {RAMB} \
  -desc {Data Width Check} -rule_body dataWidthCheck
```

The following Tcl script defines the **dataWidthCheck** procedure which is called by the **-rule_body** argument of the RAMW-1 check. This Tcl script file must be loaded into the tool using the **source** command, prior to running the report_drc command.

```
# This is a simplistic check -- report BRAM cells with WRITE_WIDTH_B wider than 36.
proc dataWidthCheck {} {
  # list to hold violations
  set vios {}
  # iterate through the objects to be checked
  foreach bram [get_cells -hier -filter {PRIMITIVE_SUBGROUP == bram}] {
    set bwidth [get_property WRITE_WIDTH_B $bram]
    if { $bwidth > 36} {
      # define the message to report when violations are found
      set msg "On cell %ELG, WRITE_WIDTH_B is $bwidth"
      set vio [ create_drc_violation -name {RAMW-1} -msg $msg $bram ]
      lappend vios $vio
    }
  }
  if {[llength $vios] > 0} {
    return -code error $vios
  } else {
    return {}
  }
}
create_drc_check -name {RAMW-1} \
                 -category {RAMB Checks} \
                 -desc {Data Width Check} \
                 -rule_body dataWidthCheck
```

Note The script file can contain both the Tcl checker procedure, and the **create_drc_check** command that defines it for use by **report_drc** command. In this case, when the Tcl script file is sourced, both the **dataWidthCheck** proc and the RAMW-1 design rule check are loaded into the tool.

See Also

- create_drc_violation
- delete_drc_check
- report_drc

create_drc_violation

Create a drc violation.

Syntax

```
create_drc_violation -name arg [-severity arg] [-msg arg]
[-quiet] [-verbose] [objects...]
```

Returns

Nothing

Usage

Name	Description
-name	Specify the name for this rule. This is the typically a 4-6 letter specification for your rule.
[**-severity**]	Specify severity level for a drc rule. Default: WARNING. Values: FATAL, ERROR, CRITICAL WARNING, WARNING, ADVISORY.
[**-msg**]	Specify your message string for this drc rule.
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*objects*]	Cells, ports, pins, nets, clock regions, sites, package banks to query.

Categories

DRC, Report

Description

Create a DRC violation object and manage the list of design objects associated with the violation for reporting by the **report_drc** command.

The **create_drc_violation** command is specified as part of the Tcl checker procedure that defines and implements the checking feature of a user-defined design rule check created by the **create_drc_check** command. A violation object is created by the Tcl checker each time a violation of the design rule is encountered.

The process in brief is:

- Write a Tcl checker procedure to define the method applied when checking the user-defined rule, and the objects to check against the rule. The Tcl checker procedure is defined in a separate Tcl script that must be loaded by the **source** command prior to running **report_drc**.

- Use **create_drc_violation** in the Tcl checker to identify and flag violations found when checking the rule against a design.

- Define a user-defined DRC rule check using the **create_drc_check** command that calls the Tcl checker proc from the **-rule_body**.

- Create a rule deck using the **create_drc_ruledeck** command, and add the user-defined rule check to the rule deck using the **add_drc_checks** command.

- Run **report_drc**, and specify either the rule deck, or the user-defined rule check to check for violations.

Violations are reported by the **report_drc** command, and violation objects can be returned by the **get_drc_vios** command.

Arguments

-name *arg* - The name of the design rule check associated with the violation. This should be the same name used by the **create_drc_check** command which calls the associated Tcl checker procedure from its **-rule_body** argument. Messages from the create_drc_violation command are passed up to the drc_check with the same **-name**.

-severity *arg* - (Optional) The severity of the violation. The default severity level for user-defined DRCs is WARNING. The supported values are:

- FATAL

- ERROR

- "CRITICAL WARNING"

- WARNING

- ADVISORY

-msg *arg* - (Optional) This is a violation specific message that is substituted for the general string variable (%STR) specified in the optional placeholder message defined in the **create_drc_check** command.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

objects - (Optional) Cell, port, pin, net, clock region, site, and package I/O bank objects associated with violations found by the Tcl checker procedure that are substituted into the placeholder message of the drc_object with the same **-name**. Design objects map to substitution keys in the message as follows:

- %ELG - netlist elements such as cells, ports, pins, and nets.

- %CRG - clock regions.

- %SIG - device sites.

- %PBG - package I/O banks.

Note Both the order and the type of *objects* passed from the create_drc_violation command must match the **-msg** specification from the **create_drc_check** command, or the expected substitution will not occur

Examples

The following Tcl script defines the **dataWidthCheck** procedure which is called by the **-rule_body** argument of the RAMW-1 check. This Tcl script file must be loaded into the tool using the **source** command, prior to running the report_drc command.

Some features of the Tcl checker proc to notice are:

- A list variable is created to store violations (**$vios**)

- A violation object is created, and added to the list variable, each time a violation is found.

- The placeholder key **%ELG** in the **$msg** string is dynamically substituted with the specific **$bram** cell associated with the violation.

- The **dataWidthCheck** proc returns an error code when any violations are found (**$vios >0**) to inform the **report_drc** command of the results of the check.

- The list of violations is passed along with the return code, and the violations are reported by **report_drc**.

```
# This is a simplistic check -- report BRAM cells with WRITE_WIDTH_B wider than 36.
proc dataWidthCheck {} {
  # list to hold violations
  set vios {}
  # iterate through the objects to be checked
  foreach bram [get_cells -hier -filter {PRIMITIVE_SUBGROUP == bram}] {
    set bwidth [get_property WRITE_WIDTH_B $bram]
    if { $bwidth > 36} {
      # define the message to report when violations are found
      set msg "On cell %ELG, WRITE_WIDTH_B is $bwidth"
      set vio [ create_drc_violation -name {RAMW-1} -msg $msg $bram ]
      lappend vios $vio
    }
  }
  if {[llength $vios] > 0} {
    return -code error $vios
  } else {
    return {}
  }
}
create_drc_check -name {RAMW-1} \
                 -category {RAMB Checks} \
                 -desc {Data Width Check} \
                 -rule_body dataWidthCheck
```

Note The script file can contain both the Tcl checker procedure, and the **create_drc_check** command that defines it for use by **report_drc** command. In this case, when the Tcl script file is sourced, both the **dataWidthCheck** proc and the RAMW-1 design rule check are loaded into the tool.

See Also

- create_drc_check
- report_drc

create_fileset

Create a new fileset.

Syntax

create_fileset [**-constrset**] [**-simset**] [**-quiet**] [**-verbose**] *name*

Returns

New fileset object

Usage

Name	Description
[**-constrset**]	Create fileset as constraints fileset (default)
[**-simset**]	Create fileset as simulation source fileset
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
name	Name of the fileset to be create

Categories

Project, Simulation

Description

Defines a new fileset within your project.

A fileset is a list 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 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** argument has been specified.

Arguments

-constrset - (Optional) 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 - (Optional) Create a simulation fileset to hold one or more simulation source files. You can only specify one type of fileset argument, either **-constrset** or **-simset**. You will get an error if both are specified.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

name - 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 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_interface

Create a new I/O port interface.

Syntax

create_interface [**-parent** *arg*] [**-quiet**] [**-verbose**] *name*

Returns

New interface object

Usage

Name	Description
[**-parent**]	Assign new interface to this parent interface
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
name	Name for new I/O port interface

Categories

PinPlanning

Description

Creates a new interface for grouping scalar or differential I/O ports.

Arguments

-parent *arg* - (Optional) Assign the new interface to the specified parent interface.

Note If the specified parent interface does not exist, an error will be returned.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

name - The name of the I/O port interface to create.

Examples

Create a new USB interface:

```
create_interface USB0
```

Create an Ethernet interface within the specified parent interface:

```
create_interface -parent Top_Int ENET0
```

See Also

- delete_interface
- create_port
- make_diff_pair_ports
- place_ports
- remove_port
- set_package_pin_val
- split_diff_pair_ports

create_ip

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

Syntax

```
create_ip [-vlnv arg] -module_name arg [-dir arg] [-vendor arg]
[-library arg] [-name arg] [-version arg] [-quiet] [-verbose]
```

Returns

List of file objects that were added

Usage

Name	Description
[**-vlnv**]	VLNV string for the Catalog IP from which the new IP will be created
-module_name	Name for the new IP that will be added to the project
[**-dir**]	Directory path for remote IP to be created and managed outside the project
[**-vendor**]	IP Vendor name
[**-library**]	IP Library name
[**-name**]	IP Name
[**-version**]	IP Version
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

IPFlow

Description

This command creates an XCI file for a configurable IP core from the IP catalog, and adds it to the source files of the current project. This creates an IP source object which must be instantiated into the HDL design to create an instance of the IP core in the netlist.

For multiple instances of the same core, simply instantiate the core module into the HDL design as many times as needed. However, to use the same IP core with different customizations, use the **create_ip** command to create separate IP source objects.

The **create_ip** command is used to import IP cores from the current IP catalog. Use the **import_ip** command to read existing XCI and XCO files directly, without having to add IP to a catalog.

This command returns a transcript of the IP generation process, concluding with the file path and name of the imported IP core file.

Note IP cores are native to Vivado, and can be customized and regenerated within that tool. However, PlanAhead provides integration to ChipScope to support legacy IP cores, and any customization and regeneration will occur within that tool. The **convert_ip** command lets you to convert legacy IP to native IP supported by Vivado.

Arguments

-vlnv *<arg>* - (Optional) Specifies the VLNV string for the existing Catalog IP from which the new IP will be created. The VLNV is the *Vendor:Library:Name:Version* string which identifies the IP in the catalog. The VLNV string maps to the IPDEF property on the IP core.

Note You must specify either **-vlnv** or all of **-vendor**, **-library**, **-name**, and **-version**

-module_name *<arg>* - Specifies the name for the new IP instance that will be added to the project

-dir *<arg>* - (Optional) The directory to write the IP core files into. If this option is not specified, the IP core files (.xci, .ngc, .veo...) are written into the hierarchy of the *<project_name>*.srcs directory.

Note This argument is only available for use in Vivado. If you specify this argument in PlanAhead, or if the specified directory does not exist, an error will be returned.

-vendor *<arg>* - (Optional) Specifies the vendor name for the IP's creator.

-library *<arg>* - (Optional) Specifies the IP library from which the core should be added.

-name *<arg>* - (Optional) Specifies the name of the IP core in the catalog.

-version *<arg>* - (Optional) Specifies the version number for the IP core.

Note You must specify either **-vlnv** or all of **-vendor**, **-library**, **-name**, and **-version**

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The example below imports the IP core specified by the **-vlnv** string, and gives it the specified module name in the current project:

```
create_ip -vlnv xilinx.com:ip:c_addsub:11.0 -module_name test_addr
```

The following example, from Vivado, creates an IP block with the specified **-vendor**, **-library**, **-name**, **-version** values, and assigns it the specified module name. After the IP is created, attributes of the IP are customized using **set_property** commands. Then the instantiation template and the synthesis targets are generated for the IP:

```
create_ip -name c_addsub -version 11.0 -vendor xilinx.com -library ip \
    -module_name c_addsub_v11_0_0
set_property -name CONFIG.Component_Name -value {c_addsub_v11_0_0} \
    -objects [get_ips c_addsub_v11_0_0]
set_property -name CONFIG.A_Width -value {32} \
    -objects [get_ips c_addsub_v11_0_0]
set_property -name CONFIG.B_Width -value {32} \
    -objects [get_ips c_addsub_v11_0_0]
set_property -name CONFIG.Add_Mode -value {Add_Subtract} \
    -objects [get_ips c_addsub_v11_0_0]
set_property -name CONFIG.C_In -value {true} \
    -objects [get_ips c_addsub_v11_0_0]
generate_target  {instantiation_template synthesis} \
    [get_files C:/Data/c_addsub_v11_0_0/c_addsub_v11_0_0.xci \
    -of_objects [get_filesets sources_1]]
```

See Also

- generate_target
- import_ip
- upgrade_ip

create_pblock

Create a new Pblock.

Syntax

create_pblock [**-parent** *arg*] [**-quiet**] [**-verbose**] *name*

Returns

New pblock object

Usage

Name	Description
[-parent]	Parent of the new pblock
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
name	Name of the new pblock

Categories

XDC, Floorplan

Description

Defines a Pblock to allow you to add logic instances for floorplanning purposes.

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 **resize_pblocks** command. The **resize_pblock** command can also be used to manually move and resize pblocks.

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.

Arguments

-parent *arg* - 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 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

name - 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_pin

Create pins in the current design.

Syntax

create_pin [**-from** *arg*] [**-to** *arg*] **-direction** *arg* [**-quiet**]
[**-verbose**] *pins*...

Returns

Nothing

Usage

Name	Description
[**-from**]	Starting bus index
[**-to**]	Ending bus index
-direction	Pin direction Values: IN, OUT, INOUT
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
pins	Names of pins to create

Categories

Netlist

Description

Add single pins or bus pins to the current netlist of an open Synthesized or Implemented Design. You may define attributes of the pin such as direction and bus width, as well as the pin name.

The pins must be created on an existing cell instance, or it is considered a top-level pin which should be created using the **create_port** command. If the instance name of a cell is not specified as part of the pin name, an error will be returned.

Netlist editing changes the in-memory view of the netlist in the current design. It does not change the files in the source fileset, or change the persistent design on the disk. Changes made to the netlist may be saved to a design checkpoint using the **write_checkpoint** command, or may be exported to a netlist file such as Verilog, VHDL, or EDIF, using the appropriate **write_*** command.

Note Netlist editing is not allowed on the elaborated RTL design.

Arguments

-bus_from *arg* - (Optional) The starting index of a bus pin.

-bus_to *arg* - (Optional) The ending index of a bus pin.

-direction [IN | OUT | INOUT] - The direction of the pin. Valid values are IN, OUT, and INOUT.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

pins - The name of the pins to create. You must specify the pin names hierarchically from the cell instance the pin is assigned to. Pins created at the top-level of the design are ports, and should be created with the **create_port** command.

Examples

The following example creates a new input pin on the cpuEngine module with the specified pin name:

```
create_pin -direction IN cpuEngine/inPin
```

The following example sets the hierarchy separator, creates a new black box instance of the reference cell, and creates a twenty-four bit bidirectional bus for that instance:

```
set_hierarchy_separator |
create_cell -reference dmaBlock -black_box usbEngine0|myDMA
create_pin -direction INOUT -bus_from 0 -bus_to 23 usbEngine0|myDMA|dataBus
```

See Also

- remove_pin
- set_hierarchy_separator
- write_edif
- write_verilog
- write_vhdl

create_port

Create scalar or bus port.

Syntax

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

Returns

List of port objects that were created

Usage

Name	Description
-direction	Direction of port. Valid arguments are IN, OUT and INOUT
[**-from**]	Beginning index of new bus
[**-to**]	Ending index of new bus
[**-diff_pair**]	Create differential pair of ports
[**-interface**]	Assign new port to this interface
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
name	Name of the port
[*negative_name*]	Optional negative name of a diff-pair

Categories

PinPlanning

Description

Creates a port and specifies such parameters as direction, width, single-ended or differential, and optionally assigns it to an existing interface. New ports are added at the top-level of the design hiearchy.

The create_port command can be used to create a new port in an I/O Planning project, or while editing the netlist of an open Synthesized or Implemented design.

Netlist editing changes the in-memory view of the netlist in the current design. It does not change the files in the source fileset, or change the persistent design on the disk. Changes made to the netlist may be saved to a design checkpoint using the **write_checkpoint** command, or may be exported to a netlist file such as Verilog, VHDL, or EDIF, using the appropriate **write_*** command.

Note Netlist editing is not allowed on the elaborated RTL design.

Arguments

-direction [IN | OUT | INOUT] - The direction of the port. Valid arguments are IN, OUT, and INOUT.

-from *arg* - (Optional) The beginning index of a new bus. A bus can start from a negative index value.

-to *arg* - (Optional) The ending index of a new bus. A bus can end on a negative index value.

-diff_pair - (Optional) Create the specified port as a differential pair of ports. In this case both a P and N port will be created for the specified port *name*.

-**interface** *arg* - (Optional) Assign the port to the specified interface.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

name - The name of the port to create.

Examples

The following example creates a new input port, named PORT0:

```
create_port -direction IN PORT0
```

The following example creates a four-bit, differential pair output bus utilizing the specified interface:

```
create_port -direction OUT -from 0 -to 3 -diff_pair -interface INTERFACE D_BUS
```

See Also

- create_interface
- make_diff_pair_ports
- place_ports
- remove_port
- write_edif
- write_verilog
- write_vhdl

create_project

Create a new project.

Syntax

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

Returns

New project object

Usage

Name	Description
[-part]	Target part
[-force]	Overwrite existing project directory
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[*name*]	Project name
[*dir*]	Directory where the project file is saved Default: .

Categories

Project

Description

Creates a project file in the specified directory.

Arguments

-part *arg* - (Optional) 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.

-force - (Optional) 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 tool to overwrite the existing project.

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

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

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 tool interprets the first argument as *name* and uses the second argument as *dir*. A project file is created *name*.`xpr` (or *name*.`ppr`), and a project data folder is also created *name*.`data` and both are written into the specified directory *dir*.

Note The project file created by the 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 - (Optional) 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 tool uses the specified path name. However, if *dir* is specified without a path, the tool looks for or creates the directory in the current working directory, or the directory from which the tool was launched.

Note When creating a project in GUI-mode, the 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.

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 tool will create the project in the current working directory, or the directory from which the tool was launched.

The following example creates a project called Proj1 in a directory called FPGA in C:/Designs. In addition, the 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**] [**-verbose**] *name class*

Returns

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

Usage

Name	Description
[**-type**]	Type of property to create; valid values are: string, int, long, double, bool Default: string
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
name	Name of property to create
class	Object type to create property for; 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* - 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 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

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

class - The class of object to assign the new property to. All objects of the specified class will be assigned the newly defined property. 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]
[-verbose] name cell
```

Returns

New reconfigurable module object

Usage

Name	Description
[**-force**]	Run the command, even if there are pending constraint changes, which will be lost
[**-blackbox**]	Create a Black Box Reconfigurable Module. Source and constraint files may not be added to a Black Box RM.
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
name	Name of the new Reconfigurable Module
cell	Cell to receive the new Reconfigurable Module

Categories

PartialReconfiguration

Description

Create a reconfigurable module (RM) and assign it to the specified cell. This defines the specified cell as reconfigurable partition, allows you to associate design sources and constraints to the partition in the sources view, and creates a PBlock containing the partition logic.

A single partition can have multiple RMs to contain different netlists, constraints, or implementations. Use the **set_property** command to define the design source for the RM fileset, using the **edif_top_file** property. See the example below.

You can also define a blackbox RM for the partition cell in order to blackout the contents of the partition as needed.

Use the **load_reconfig_module** command to make a specific module active for a partition cell. The combination of active modules in the design, and all other logic, is called a configuration of the design.

A design with multiple configurations should be checked to insure that the static logic and partition pins are consistent across all configurations. You can use the **verify_config** command to check the configurations of a design.

This command returns the hierarchical name of the Tcl newly created RM.

Arguments

-force - (Optional) Force the creation of the RM even when there are outstanding design changes to be saved. If there are pending design changes, and **-force** is not specified, an error will be returned.

Note You should use **save_design** prior to using **create_reconfig_module**.

-blackbox - (Optional) Create a Black Box RM. Design source and constraint files may not be added to a Black Box RM.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

name - The name of the new RM to create. This name is added to the cell name to create the hierarchical name of the RM *<cell>:<name>*.

cell - The name of the partition cell to assign the RM to.

Examples

The example below creates a reconfigurable module with the name BramFirst on the specified cell, sets the **edif_top_file** property to reference the appropriate netlist file, and loads the module netlist into the current design:

```
create_reconfig_module -name BramFirst -cell U1_RP_Bram
set_property edif_top_file \
   C:/Data/xpr_bram_led/Synth/BramFirst/recon_block_bram.ngc \
   [get_filesets U1_RP_Bram#BramFirst]
save_design
load_reconfig_modules -reconfig_modules U1_RP_Bram:BramFirst
```

Note The name of the module fileset specified in the **set_property** command is the concatenated names of the cell and the reconfigurable module name: *<cell>#<name>*

The example below creates a Blackbox RM named Bram_BB on the specified cell:

```
create_reconfig_module -blackbox -name Bram_BB -cell U1_RP_Bram
```

See Also

- config_partition
- create_pblock
- delete_reconfig_module
- set_property
- verify_config

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] [-verbose] name
```

Returns

Run object

Usage

Name	Description
[-constrset]	Constraint fileset to use
[-parent_run]	Synthesis run to link to new implementation run
[-part]	Target part
-flow	Flow name
[-strategy]	Strategy to apply to the run
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
name	Name for new run

Categories

Project

Description

Defines a synthesis or implementation run. The attributes of the run can be configured with the use of the **set_property** command.

Arguments

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

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

-part *partName* - 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* - The tool flow and release version for the synthesis tool {XST 14} or the implementation tool {ISE 14}.

-strategy *arg* - The strategy to employ for the synthesis or implementation run. There are many different strategies to choose from within the tool, including custom strategies you can define. Refer to the appropriate user guide 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

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

Examples

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

```
create_run -flow {XST 14} 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 14 tool flow, and attaches it to the first_pass synthesis run previously created:

```
create_run -flow {ISE 14} –parent_run first_pass impl_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] [-verbose]
```

Returns

Nothing

Usage

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

Categories

[Report](#)

Description

Create a slack histogram grouping paths into slack ranges, and displaying the results graphically.

This command provides a graphical slack histogram that requires the tool to be running in GUI mode the **-name** argument to be used.

Arguments

-to *args* - (Optional) Specify a clock name, to analyze paths that end in the specified clock domain.

-delay_type *arg* - (Optional) Specifies the type of path delay to analyze when creating the slack report. The valid values are min, max, and min_max. The default setting for **-delay_type** is max.

-num_bins *args* - (Optional) Specify the number of slack bins to divide the results into. The number of bins determines the granularity of the histogram returned. The range of slack values calculated is divided evenly into the specified number of bins, and the paths are grouped into the bins according to their slack values. The value can be specified as a number => 1, with a default value of 10.

-slack_less_than *arg* - Report slack on paths with a calculated slack value less than the specified value. Used with **-slack_greater_than** to provide a range of slack values of specific interest.

-slack_greater_than *arg* - Report slack on paths with a calculated slack value greater than the specified value. Used with **-slack_less_than** to provide a range of slack values of specific interest.

-group *args* - Report slack for paths in the specified path groups. Currently defined path groups can be determined with the **get_path_groups** command.

-report_unconstrained - Report delay slack on unconstrained paths. By default, unconstrained paths are not analyzed.

-significant_digits *arg* - The number of significant digits in the output results. The valid range is 0 to 3. The default setting is 3 significant digits.

-scale [linear | logarithmic] - Specify the Y-axis scale to use when presenting the slack histogram. Logarithmic allows for a smoother presentation of greatly different values, but linear is the default.

-name *arg* - (Optional) Specifies the name of the results set for the GUI. If the name specified is currently opened, the **create_slack_histogram** will overwrite the current results.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example creates a slack histogram of the current design, using the default values, and outputting the results to the named result set in the GUI:

```
create_slack_histogram -name slack1
```

See Also

- delete_timing_results
- get_path_groups
- report_timing

create_sysgen

Create DSP source for Xilinx System Generator and add to the source fileset.

Syntax

create_sysgen [**-quiet**] [**-verbose**] *name*

Returns

Name for the new sub module

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
name	Sub module name

Categories

SysGen

Description

Create a DSP sub-module for use in the current project, and add it to the source files.

This command will launch System Generator for DSP to let you design the hardware portion of the embedded processor system. System Generator is a DSP design tool from Xilinx that allows the RTL source files, Simulink® and MATLAB® software models, and C/C++ components of a DSP system to come together in a single simulation and implementation environment.

For more information on using specific features of the tool refer to *System Generator for DSP Getting Started Guide* (UG639).

You can also add existing DSP model files (.mdl) from System Generator into the current project using the **add_files** command.

The command returns the name of the DSP module created and added to the project.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

name - The name of the DSP module to create and add to the current project.

Examples

The following example launches System Generator and allows you to define and configure the specified DSP module:

```
create_sysgen DSP_mod1
```

See Also

- add_files
- generate_target
- list_targets
- make_wrapper

create_xps

Create embedded source for XPS and add to the source fileset.

Syntax

create_xps [**-quiet**] [**-verbose**] *name*

Returns

Source file name that was created

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
name	Source name

Categories

[XPS](#)

Description

Create an Embedded Processor source for use in the current project, and add it to the source files.

This command will launch the Xilinx Platform Studio (XPS) to let you design the hardware portion of the embedded processor system. In XPS you can define and configure the microprocessor, peripherals, and the interconnection of these components. After you exit XPS, the created files for the Embedded Processor sub-design will be written to the local project directory (*project_name*.srcs/sources_1/edk/*name*), and added to the source files.

For more information on using specific features of XPS refer to *EDK Concepts, Tools, and Techniques* (UG683).

You can also add existing Xilinx Microprocessor Project (.xmp) files from XPS in the current project using the **add_files** command.

The command returns the name of the Embedded Processor sub-design created.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

name - The name of the Embedded Processor sub-design to create and add to the current project.

Examples

The following example launches XPS to define and configure the specified Embedded Processor sub-design:

```
create_xps xpsTest1
```

See Also

- add_files
- generate_target
- list_targets
- make_wrapper

crossprobe_fed

Crossprobe paths of Bels and Nets to FPGAEditor.

Syntax

crossprobe_fed [**-run** *arg*] [**-path** *args*] [**-objects** *args*] [**-quiet**]
[**-verbose**]

Returns

Nothing

Usage

Name	Description
[**-run**]	Implemented run to launch FED with
[**-path**]	Path connecting primitive cells and nets
[**-objects**]	List of cells and nets to crossprobe
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

ToolLaunch

Description

This command allows you to cross-probe from the 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.

Arguments

-run *name* - (Optional) The run name to use when cross-probing.

-path *paths* - (Optional) One or more paths to cross-probe in the FPGA Editor.

-objects *objects* - (Optional) 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Chapter 3: **Tcl Commands Listed Alphabetically**

XILINX®

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_fifo/infer_fifo.empty_reg_reg}
```

See Also

- get_cells
- get_ports
- launch_fpga_editor

current_design

Set or get the current design.

Syntax

current_design [**-quiet**] [**-verbose**] [*design*]

Returns

Design object

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*design*]	Name of current design to be set

Categories

SDC, XDC

Description

Defines the current design or returns 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 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.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

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

Examples

The following example sets the design rtl_1 as the current design:

```
current_design rtl_1
```

See Also

- current_instance
- get_designs
- get_projects

current_fileset

Set or get the current fileset.

Syntax

```
current_fileset [-constrset] [-simset] [-quiet] [-verbose]
[fileset]
```

Returns

Current fileset (the current srcset by default)

Usage

Name	Description
[**-constrset**]	Get the current constraints fileset
[**-simset**]	Get the current active simulation fileset
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*fileset*]	Fileset to set as active; optional

Categories

Project

Description

Get or set the active source, constraint, or simulation fileset within the current project.

This command returns the current fileset as specified. When used without any options, the current_fileset sets and returns the sources_1 set as the active fileset.

Arguments

-constrset - (Optional) Set or return the currently active constraint set.

-simset - (Optional) Set or return the currently active simulation fileset.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

fileset - (Optional) The name of the fileset to make active. This argument sets the active simulation or constraints fileset in projects with multiple filesets. When *fileset* is not specified, the sources_1 fileset is returned as the active fileset.

Examples

The following example returns the name of the currently active simulation file set:

```
current_fileset -simset
```

The following example sets `constrs_2` as the active constraint set:

```
current_fileset constrs_2
```

See Also

- create_fileset
- delete_fileset
- get_filesets

current_instance

Set or get the current instance.

Syntax

current_instance [**-quiet**] [**-verbose**] [*instance*]

Returns

Instance name

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*instance*]	Name of instance

Categories

SDC, XDC

Description

Set the current instance in the design hierarchy to the specified instance cell or to the top module. By default, **current_instance** points to the top module of the **current_design**, which is not an instantiated cell object. You can also set **current_instance** to reference an instantiated hierarchical cell in the design.

Since the top module is not an instantiated object, this command returns a string with the name of the current instance, rather than an object.

The current design and current instance are the target of most of the commands and design changes you will make. 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. You can determine the current hierarchy separator with the **get_hierarchy_separator** command.

Use '..' to traverse up the hierarchical instance path when specifying the current instance.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

instance - (Optional) The name of the instance to be set as the current instance of the current design.

- The *instance* is specified relative to the presently defined current instance, using the defined hierarchy separator.

- Use '..' to move up one level of the hierarchy relative to the current instance.

- If the *instance* argument is omitted, the current instance is reset to the top module in the design hierarchy.

- If the *instance* is specified as '.' then the name of the current instance is returned, and the instance is not changed.

Examples

The following example sets the current instance to the top module of the current design:

```
current_instance
   INFO: [PlanAhead 12-618] Current instance is the top level of design 'netlist_1'.
   top
```

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**] [**-verbose**] [*project*]

Returns

Current or newly set project object

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*project*]	Project to set as current

Categories

Project

Description

Specifies the current project or returns the current project when no project is specified.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

project - (Optional) The name of the 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.

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 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 tool:

```
current_project
```

Note The returned value is the name of the 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**] [**-verbose**]
[*run*]

Returns

Run object

Usage

Name	Description
[**-synthesis**]	Set or get the current synthesis run
[**-implementation**]	Set or get the current implementation run (default unless '-synthesis' is specified)
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*run*]	Run to set as current; optional

Categories

Project

Description

Defines the current synthesis or implementation run, or returns 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 - (Optional) Specifies that the **current_run** command should set or return the name of the current synthesis run.

-implementation - (Optional) 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

run - (Optional) Sets 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 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

delete_debug_core

Delete ChipScope debug core.

Syntax

delete_debug_core [**-quiet**] [**-verbose**] *cores...*

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
cores	ChipScope debug cores to delete

Categories

ChipScope

Description

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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

cores - 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**] [**-verbose**] *ports...*

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
ports	ChipScope debug ports to delete

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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

ports - 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
```

Note 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_drc_check

Delete one or more user-defined drc checks.

Syntax

```
delete_drc_check [-quiet] [-verbose] name ...
```

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
name	Specify the key for the check to remove. This is the typically of the form PREFIX-id where PREFIX is a 4-6 letter abbreviation and id is a unique identifier. Use get_drc_checks to determine the correct name to use. Only user-defined rules may be deleted.

Categories

DRC, Object

Description

Delete a single user-defined design rule checks from the current project. User-defined design rule checks are created using the **create_drc_checks** command.

Note You cannot delete factory defined rule checks.

Once it has been deleted there is no way to recover a rule check. The undo command will not work.

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

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

name - Specify the name of a user-defined design rule check to be deleted from the current project.

Examples

The following example deletes the specified design rule check:

```
delete_drc_check LJH-1
```

See Also

create_drc_check

delete_fileset

Delete a fileset.

Syntax

delete_fileset [**-quiet**] [**-verbose**] *fileset*

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
fileset	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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

fileset - The name of the fileset to delete. The last constraint or simulation fileset will not be deleted, and no error will be returned under these circumstances.

Examples

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**] [**-verbose**] *interfaces*...

Returns

Nothing

Usage

Name	Description
[**-all**]	Also remove all of the ports and buses belonging to the interface
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
interfaces	I/O port interfaces to remove

Categories

[PinPlanning](#)

Description

Deletes an existing interface and optionally deletes all of the associated ports and buses using the interface.

Arguments

-all - (Optional) Delete all ports, buses, or nested interfaces associated with the specified interface.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

interfaces - The name of interfaces to delete.

Examples

The following example deletes the specified interface and all of its associated ports and buses:

```
delete_interface USB0
```

See Also

- create_interface
- create_port

delete_pblock

Remove Pblock.

Syntax

delete_pblock [**-hier**] [**-quiet**] [**-verbose**] *pblocks*...

Returns

Nothing

Usage

Name	Description
[**-hier**]	Also delete all the children of Pblock
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
pblocks	Pblocks to delete

Categories

Floorplan, XDC

Description

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

Arguments

-hier - (Optional) 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

pblocks - One or more Pblocks to be deleted.

Examples

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

```
delete_pblock -hier cpuEngine
```

See Also

create_pblock

delete_power_results

Delete power results that were stored in memory under a given name.

Syntax

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

Returns

Nothing

Usage

Name	Description
-name	Name for the set of results to clear
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Power

Description

Deletes 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* - The name of the results set to delete. This name was either explicitly defined, or was automatically defined when the **report_power** command was run.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

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**] [**-verbose**] *reconfig_module*

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
reconfig_module	Reconfigurable Module(s) to delete

Categories

PartialReconfiguration

Description

Removes the specified reconfigurable module (RM) from the current design.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

reconfig_module - One or more reconfiurable modules to delete from the design. The RM is specified by the hierarchical name *<cell>:<name>*.

Examples

In the example below the BramSecond reconfigurable module is deleted from the U1_RP_Bram partition cell:

```
delete_reconfig_module U1_RP_Bram:BramSecond
```

See Also

create_reconfig_module

delete_rpm

Delete an RPM.

Syntax

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

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
rpm	RPM to delete

Categories

Floorplan

Description

Deletes the specified Relationally Placed Macro (RPM) from the design.

An RPM is a list 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

rpm - The RPM to be deleted.

Examples

The following example deletes the specified RPM (cs_ila_0/U0) from the design:

```
delete_rpm cs_ila_0/U0
```

delete_run

Delete an existing run.

Syntax

delete_run [**-noclean_dir**] [**-quiet**] [**-verbose**] *run*

Returns

Nothing

Usage

Name	Description
[**-noclean_dir**]	Do not remove all output files and directories from disk
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
run	Run to modify

Categories

Project

Description

Deletes the specified run from the project, and deletes all results of the run from the project directory on the hard drive unless otherwise specified.

Arguments

-noclean_dir - Do 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

run - 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**] [**-verbose**] *name*

Returns

Nothing

Usage

Name	Description
[**-type**]	Type of timing results to clear; Values: timing_path, slack_histogram, clock_interaction, check_timing, pulse_width, timing_summary
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
name	Name for the set of results to clear

Categories

Report, Timing

Description

Clear the specified timing results from the named result set. Both the type of the timing report, and the name of the timing report must be specified, or the command will fail.

Arguments

-type *arg* - (Optional) Specifies the type of timing results to be cleared. The available types are:

- timing_path - Delete the named **report_timing** report.
- slack_histogram - Delete the named **create_slack_histogram** report.
- clock_interaction - Delete the named **report_clock_interaction** report.
- check_timing - Delete the named **check_timing** report.
- pulse_width - Delete the named **report_pulse_width** report.
- timing_summary - Delete the named **report_timing_summary** report.

Note The default **-type** is timing_path, to delete reports generated by the **report_timing** command.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

-name *arg* - Specifies the name of the timing results to be cleared.

Examples

The following example clears the specified results set from memory:

```
delete_timing_results -type clock_interaction -name clkNets
```

See Also

- create_slack_histogram
- report_clock_interaction
- report_pulse_width
- report_timing

delete_utilization_results

Delete utilization results that were stored in memory under a given name.

Syntax

delete_utilization_results -name *arg* [**-quiet**] [**-verbose**]

Returns

Nothing

Usage

Name	Description
-name	Name for the set of results to clear
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Report

Description

Clear the specified utilization results from the named result set.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

-name *arg* - Specifies the name of the results to be cleared.

Examples

The following example clears the specified results set from memory:

```
delete_utilization_results -name SSO1
```

See Also

report_utilization

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] [-verbose]
```

Returns

Nothing

Usage

Name	Description
[-run]	Promoted run to be demoted
[-partition_names]	List of Partitions to be promoted
[-promote_dir]	Directory to be demoted
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

PartialReconfiguration, Partition

Description

Delete previously promoted partitions so they are no longer available for import using design preservation.

WARNING - This command will delete the specified partition directory and data from the hard drive. Be sure this is the desired result prior to executing this command.

Arguments

-run *args* - (optional) The run to be demoted.

-partition_names *args* - (optional) The names of partitions to be demoted.

-promote_dir *arg* - (optional) The path to the partition data to be demoted.

Note If the path to the directory is not specified, the tool will look in your home directory for the partition data:

• For Windows: %APPDATA%/Xilinx/PlanAhead
• For Linux: $HOME/.Xilinx/PlanAhead

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example demotes the specified partition data for the usbEngine0 partition:

```
demote_run -promote_dir C:/Data/partition/partition_DP_RTL.promote/Ximpl_1 \
-partition_names usbEngine0
```

See Also

- promote_run
- verify_config

disconnect_debug_port

Disconnect nets and pins from debug port channels.

Syntax

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

Returns

Nothing

Usage

Name	Description
[**-channel_index**]	Disconnect the net at channel index
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
port	Debug port name

Categories

[ChipScope](#)

Description

Disconnect signals from the debug ports.

Signals from the Netlist Design are connected to ports of a ChipScope debug core using the **connect_debug_port** command.

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, 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* - The channel index of the port to disconnect.

Note The entire port is disconnected if **channel_index** is not specified.

-**quiet** - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

port - 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.

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**] [**-verbose**]

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

GUIControl

Description

Ends a sequence of commands that can be undone or redone as a series. Use **startgroup** to start the sequence of commands.

Note You can have multiple command groups to **undo** or **redo**, but you cannot nest command groups. You must use **endgroup** to end a command sequence before using **startgroup** to create a new command sequence

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

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

- startgroup
- redo
- undo

export_hardware

Export system hardware platform for SDK.

Syntax

```
export_hardware [-bitstream] [-dir arg] [-quiet]
[-verbose] files [run]
```

Returns

Nothing

Usage

Name	Description
[**-bitstream**]	Export bitstream data to SDK export directory
[**-dir**]	Export directory
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
files	Source files for which the hardware data needs to be exported
[*run*]	Current implementation run

Categories

[XPS](#)

Description

Export the Embedded Processor system hardware platform for use by SDK to support the design of software for the embedded processor sources in your project. Specify the embedded processor XPS project files to export to SDK.

As a default, the tool will write the hardware specification file (.xml) for the specified embedded processors to the *project_name*.sdk/SDK/SDK_Export/hw directory, to a file named after the embedded processor in the design, with the .XML extension.

This is a copy of the system.xml file in the embedded processor directory of the project design sources located at: *project_name*.srcs/sources_1/edk/robot/__xps/. The system.xml file contains a description of the embedded processor, and the components of the XPS design.

The output of the **export_hardware** command can be redirected to a user-defined directory with the **-dir** option. The output of the command can be used to invoke SDK with the **launch_sdk** command.

The command returns a transcript of the export process.

Arguments

-bitstream - (Optional) Export the bitstream and BMM model data for the Embedded Processor, in addition to the hardware specification file.

Note The tool will return an error if the bitstream file does not exist

-dir *arg* - (Optional) Export directory. By default the hardware files will be written to the local project directory, under *project_name*.**sdk/SDK/SDK_Export/hw**.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

files - A files object that contains the list of XPS project files (.xmp) to export.

Note Use **get_files** to specify a files object, rather than specifying a file name.

run - (Optional) Specify an implementation run to export.

Examples

The following example exports the Embedded Processor design to the standard SDK export directory, and includes the Bitstream and BMM model data:

```
export_hardware -bitstream [get_files *.xmp]
```

See Also

- create_xps
- generate_target
- get_files
- launch_sdk

filter

Filter a list, resulting in new list.

Syntax

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

Returns

New list

Usage

Name	Description
[**-regexp**]	Operators =~ and !~ use regular expressions
[**-nocase**]	Perform case-insensitive matching (valid only when -regexp specified)
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*objects*]	List of objects to filter
[*filter*]	Filter list with expression

Categories

Object, PropertyAndParameter, XDC

Description

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

Arguments

-regexp - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

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

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

objects - 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 - The expression to use for filtering. The specified pattern filters the list of objects returned based on property values on the objects. You can find out which 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 filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

Examples

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

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

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

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

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**] [**-verbose**]

Returns

Nothing

Usage

Name	Description
[**-fileset**]	Fileset to parse to search for top candidates
[**-files**]	Files to parse to search for top candidates
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Project

Description

Find the most likely candidates for the top module in the files defined in the current fileset, or in the specified fileset, or in the specified list of files.

The command returns an ordered list of modules that the tool identifies as the best candidates for the top-level of the design. You can use the **lindex** command, and choose index 0 to select the best candidate for the top module.

Arguments

-fileset arg - (Optional) Search the specified simulation or source fileset for top module candidates. The default is to search the current fileset of the current design.

-files arg - (Optional) Find the top module candidates in the specified list of files.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example chooses the best top module of the current design for synthesis:

```
synth_design -top [lindex [find_top] 0]
```

Note Since **find_top** returns multiple possible candidates, choosing index 0 chooses the best top candidate for synthesis.

The following example returns the best top module candidate from the specified list of files:

```
find_top -files [get_files -filter {NAME =~ *or1200*}]
```

The following example sets the results of **find_top** into the variable **$topVar**, then uses that variable to define the top module in the current fileset using the **set_property** command:

```
set topVar [ find_top -files [get_files -filter {NAME =~ *usbf*} ] ]
usbf_top
set_property top $topVar [current_fileset]
```

See Also

set_property

generate_target

Generate target data for the specified source.

Syntax

generate_target [**-force**] [**-quiet**] [**-verbose**] *name objects*

Returns

Nothing

Usage

Name	Description
[**-force**]	Force target data regeneration
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
name	List of targets to be generated, or 'all' to generate all supported targets
objects	The objects for which data needs to be generated

Categories

Project, XPS, IPFlow, IPIntegrator

Description

This command generates target data for the specified source file for IP cores (.xco), DSP modules (.mdl), or Embedded Processor sub-designs (.xmp). The target data that is generated are the files necessary to support the IP core, DSP module, or Embedded Processor, through the FPGA design flow.

Legacy IP, available in the PlanAhead tool, supports only instantiation_template and synthesis targets.

For DSP modules and Embedded Processor sub-designs, Synthesis, Simulation, and Implementation are the standard targets.

You can use the **list_targets** command to list the targets supported by a specific source file.

Arguments

-force - (Optional) Force target data regeneration, and overwrite any existing target data files. Without **-force**, the tool will not regenerate any target data that is up-to-date.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

name - The names of the types of target data to create for the specified source. The specific targets supported by an IP core are listed in the SUPPORTED_TARGETS property on the object. You can query this property to see which targets a specific core supports. Standard values are:

- **all** - Generate all targets for the specified core.
- **instantiation_template** - Generate the Instantiation template used to add the RTL module definition for the IP core into the current design. The instantiation template can be copied into any desired level of the design hierarchy.
- **synthesis** - Synthesis targets deliver HDL files that are used during synthesis for native IP, or deliver a synthesized netlist file (NGC) generated by XST.

objects - The object to generate the target from. Supported objects can include IP core objects, or the IP source files (XCO), DSP modules (MDL) imported from System Generator, and Embedded Processors (XMP) imported from Xilinx Platform Studio (XPS).

Note Use **get_files** to specify a files object, rather than specifying a file name

Examples

The following example generates the implementation template for all of the IP cores in the current project, forcing regeneration of any targets which are up-to-date:

```
generate_target instantiation_template [get_ips] -force
```

The following example queries the specified IP object to report the SUPPORTED_TARGETS property, and then generates the Example target data:

```
report_property -all [get_ips blk_mem*]
generate_target {example} [get_ips blk_mem*]
```

See Also

- add_files
- create_ip
- create_sysgen
- create_xps
- import_ip
- list_targets
- open_example_project
- read_ip
- report_property
- reset_target

get_bel_pins

Get a list of bel_pins.

Syntax

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

Returns

Bel_pindef

Usage

Name	Description
[**-regexp**]	Patterns are full regular expressions
[**-nocase**]	Perform case-insensitive matching. (valid only when -regexp specified)
[**-filter**]	Filter list with expression
[**-of_objects**]	Get the bel_pindef of these bels, or pins.
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*patterns*]	Match bel_pindef against patterns Default: *

Categories

Object

Description

Returns a list of pins on the specified BELs, or matching a specified search pattern.

The default command gets a list of all pins on all BELs on the device.

Note To improve memory and performance, the get_* commands return a container list of a single type of objects (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

-nocase - (Optional) 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_bel_pins** based on property values on the pins. You can find the properties 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 PIN object, "NAME" and "IS_INVERTED" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects *args* - This option can be used with the **get_bels** command to return the pins of specified BELs.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - Match BEL pins against the specified patterns. The default pattern is the wildcard '*' which gets a list of all BEL pins on the device. More than one search pattern can be specified to find pins 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 pins of the specified BELs associated with the specified range of sites on the device:

```
get_bel_pins -of_objects [get_bels -of_objects [get_sites \
    -range {SLICE_X0Y0 SLICE_X1Y1}] ]
```

The following example returns the clock enable (CE) pins of of all BELs on the device:

```
get_bel_pins *CE
```

See Also

- get_bels
- get_sites
- list_property
- report_property

get_bels

Get a list of bels.

Syntax

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

Returns

Bels

Usage

Name	Description
[-**regexp**]	Patterns are full regular expressions
[-**nocase**]	Perform case-insensitive matching. (valid only when -regexp specified)
[-**filter**]	Filter list with expression
[-**of_objects**]	Get the bels of these sites cells clock_regions.
[-**quiet**]	Ignore command errors
[-**verbose**]	Suspend message limits during command execution
[*patterns*]	Match bels against patterns Default: *

Categories

Object

Description

Basic Elements, or BELs, are building blocks of logic, such as flip-flops, LUTs, and carry logic, that make up a SLICE. This command returns a list of BELs on the target part that match a specified search pattern in an open design.

The default command gets a list of all BELs on the device.

Note To improve memory and performance, the get_* commands return a container list of a single type of objects (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

-nocase - (Optional) 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_bels** based on property values on the BELs. You can find the properties 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 BEL object, "IS_OCCUPIED" and "TYPE" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects *args* - This option can be used with the **get_sites** command to return the BELs of specified site objects.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - Match BELs against the specified patterns. The default pattern is the wildcard '*' which gets a list of all BELs on the device. More than one search pattern can be specified to find BELs 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 total number of BELs on the target part:

```
llength [get_bels]
```

The following example returns the BELs associated with the specified site:

```
get_bels -of_objects [get_sites PHASER_IN_PHY_X0Y5]
```

See Also

- get_sites
- list_property
- report_property

get_boords

Get the list of boards available in the project.

Syntax

get_boards [**-filter** *arg*] [**-quiet**] [**-verbose**]

Returns

List of boards objects

Usage

Name	Description
[**-filter**]	Filter list with expression
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Object, Project, XPS

Description

Gets a list of evaluation boards available for the current project.

Note To improve memory and performance, the get_* commands return a container list of a single type of objects (e.g. cells, nets, pins, or 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

-filter *args* - Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by **get_boards** based on property values on the boards. You can find the properties 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 board object, "NAME", "DEVICE", and "FAMILY" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example reports the properties of the specified evaluation board:

```
report_property [get_boards -filter {NAME==ml505}]
```

See Also

- list_property
- report_property

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]
[-verbose] [patterns]
```

Returns

List of cell objects

Usage

Name	Description
[**-hsc**]	Hierarchy separator Default: /
[**-hierarchical**]	Search level-by-level in current instance
[**-regexp**]	Patterns are full regular expressions
[**-nocase**]	Perform case-insensitive matching (valid only when -regexp specified)
[**-filter**]	Filter list with expression
[**-of_objects**]	Get cells of these pins, timing paths, nets, bels or sites
[**-match_style**]	Style of pattern matching Default: sdc Values: ucf, sdc
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*patterns*]	Match cell names against patterns Default: *

Categories

SDC, XDC, Object

Description

Gets 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 (e.g. cells, nets, pins, or 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* - (Optional) Set the hierarchy separator. The default hierarchy separator is '/'.

-hierarchical - (Optional) Get cells from all levels of the design hierarchy. Without this argument, the command will only get cells from the top of the design hierarchy. When using **-hierarchical**, the search pattern should not contain a hierarchy separator because the search pattern is applied at each level of the hierarchy, not to the full hierarchical cell name. For instance, searching for U1/* searches each level of the hierarchy for instances with U1/ in the name. This may not return the intended results. This is illustrated in the examples below.

Note When used with **-regexp**, the specified search string is matched against the full hierarchical name, and the U1/* search pattern will work as intended

-regexp - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

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

-filter *args* - (Optional) 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 the properties 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 filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

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

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

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - (Optional) Match cells against the specified patterns. The default pattern is the wildcard '*' which gets a list of 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 gets a list of properties and property values attached to the second object of the list returned by get_cells:

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

Note If there are no cells matching the pattern you will get a warning.

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==1}]]]  {puts $cell}
```

The following example demonstrates the effect of **-hierarchical** searches, without and with **-regexp**:

```
get_cells -hierarchical *mmcm*
   mmcm_replicator_inst_1
   mmcm_replicator_inst_1/mmcm_stage[0].mmcm_channel[0].mmcm
get_cells -hierarchical -regexp .*mmcm.*
   mmcm_replicator_inst_1
   mmcm_replicator_inst_1/mmcm_stage[0].mmcm_channel[0].mmcm
   mmcm_replicator_inst_1/mmcm_stage[0].mmcm_channel[0].mmcm/GND
   mmcm_replicator_inst_1/mmcm_stage[0].mmcm_channel[0].mmcm/MMCM_Base
```

Note The last two cells (GND and MMCM_Base) were not returned in the first example (without **-regexp**) because the cell names do not match the search pattern, as it is applied to each level of the hierarchy.

See Also

- get_lib_cells
- get_nets
- get_pins
- list_property
- report_property

get_clock_regions

Get the clock regions for the current device.

Syntax

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

Returns

Clock_regions

Usage

Name	Description
[-regexp]	Patterns are full regular expressions.
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified).
[-filter]	Filter list with expression
[-of_objects]	Get the clock_regions of these sites
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[patterns]	Match objects' name against patterns. Default: *

Categories

Object

Description

Gets a list of clock regions on the target part that match a specified search pattern. The default command gets a list of all clock regions on the device in an open design.

Note To improve memory and performance, the get_* commands return a container list of a single type of objects (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

-nocase - (Optional) 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_clock_regions** based on property values on the clock regions. You can find the properties 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 clock region object, "COLUMN_INDEX", "HIGH_X", and "LOW_X" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects *args* - This option can be used with the **get_sites** command to return the clock region that the specified site is found in.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - Match clock regions against the specified patterns. The default pattern is the wildcard '*' which gets a list of all clock regions on the device. More than one search pattern can be specified to find clock regions 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 clock regions matching the search pattern:

```
get_clock_regions X0*
```

The following example returns the clock regions filtered by the specified property:

```
get_clock_regions -filter {LOW_X==0}
```

Note These two examples return the same set of clock regions

See Also

- get_sites
- 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] [-include_generated_clocks] [-quiet]
[-verbose] [patterns]
```

Returns

List of clocks

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get clocks of these pins or nets
[-match_style]	Style of pattern matching, valid values are ucf, sdc Default: sdc
[-include_generated_clocks]	Include auto-inferred/generated_clocks also.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[*patterns*]	Match clock names against patterns Default: *

Categories

SDC, XDC, Object

Description

Gets a list of clocks in the current design that match a specified search pattern. The default command gets 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 (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

-nocase - (Optional) 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 the properties 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 filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-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 clocks, including generated clocks that match the specified pattern 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - Match clocks against the specified patterns. The default pattern is the wildcard '*' which gets 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 gets a list of clocks matching the various search patterns:

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

Note If there are no clocks matching the pattern you will get a warning.

The following example gets the master clock object, and all generated clocks derived from that clock:

```
get_clocks -include_generated_clocks wbClk
```

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

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

See Also

- all_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] [-of_objects args] [-regexp]
[-nocase] [-quiet] [-verbose] [patterns]
```

Returns

List of debug_core objects

Usage

Name	Description
[**-filter**]	Filter list with expression
[**-of_objects**]	Get cores of these debug ports or nets
[**-regexp**]	Patterns are full regular expressions
[**-nocase**]	Perform case-insensitive matching (valid only when -regexp specified)
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*patterns*]	Match debug cores against patterns Default: *

Categories

Object, ChipScope

Description

Gets a list of ChipScope debug cores in the current project that match a specified search pattern. The default command gets 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 (e.g. cells, nets, pins, or 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

-filter *args* - (Optional) 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 the properties on an object with the **report_property** or **list_property** commands.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects *args* - Get the ChipScope debug cores associated with the specified debug ports, or nets.

-regexp - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

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

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - Match debug cores against the specified patterns. The default pattern is the wildcard '*' which gets all debug cores. More than one pattern can be specified to find multiple debug cores 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 gets 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 tool when you add any ILA cores to the project.

The following example gets 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 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_ports
- 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*] [**-of_objects** *args*] [**-regexp**]
[**-nocase**] [**-quiet**] [**-verbose**] [*patterns*]

Returns

List of debug_port objects

Usage

Name	Description
[**-filter**]	Filter list with expression
[**-of_objects**]	Get ports of these debug cores
[**-regexp**]	Patterns are full regular expressions
[**-nocase**]	Perform case-insensitive matching (valid only when -regexp specified)
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*patterns*]	Match debug ports against patterns Default: *

Categories

Object, ChipScope

Description

Gets a list of ports defined on ChipScope debug cores in the current project that match a specified search pattern. The default command gets a list of all debug ports in the project.

Debug ports are defined when ChipScope debug cores are created with the **read_chipscope_cdc** command, or the **create_debug_core** command. 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 (e.g. cells, nets, pins, or 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

-filter *args* - (Optional) Filter the results list with the specified expression. The **-filter** argument filters the list of objects returned by **get_debug_ports** based on property values on the ports. You can find the properties 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 filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects *args* - Get the ChipScope debug ports associated with the specified debug cores.

-regexp - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

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

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - Match debug ports against the specified patterns. The default pattern is the wildcard '*' which gets all debug ports. More than one pattern can be specified to find multiple debug ports 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 gets 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 gets 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_delays

Returns delay objects.

Syntax

```
get_delays [-regexp] [-nocase] [-filter arg] [-of_objects args]
[-quiet] [-verbose]
```

Returns

Nothing

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get the delays of the objects.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Object

get_designs

Get a list of designs in the current design.

Syntax

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

Returns

List of design objects

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[*patterns*]	Match design names against patterns Default: *

Categories

Object

Description

Gets a list of open designs in the current project that match a specified search pattern. The default command gets 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 (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

-nocase - (Optional) 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 the properties on an object with the **report_property** or **list_property** commands. In the case of the "design" object, "CONSTRSET", and "PART" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

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

Examples

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

```
get_designs
```

The following example gets 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 you will get a warning.

See Also

report_property

get_files

Get a list of source files.

Syntax

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

Returns

List of file objects

Usage

Name	Description
[**-regexp**]	Patterns are full regular expressions
[**-nocase**]	Perform case-insensitive matching (valid only when -regexp specified)
[**-filter**]	Filter list with expression
[**-of_objects**]	Get files of these filesets or composite files
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*patterns*]	Match file names against patterns Default: *

Categories

Object, Project

Description

Gets a list of files in the current project that match a specified search pattern. The default command gets 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 (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

-nocase - (Optional) 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 the properties 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 "file" object, "FILE_TYPE", and "IS_ENABLED" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

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

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - Match files against the specified patterns. The default pattern is the wildcard '*' which gets 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 the Verilog files in the design:

```
get_files -filter {FILE_TYPE == Verilog}
```

The following example gets 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 you will get a warning.

See Also

report_property

get_filesets

Get a list of filesets in the current project.

Syntax

get_filesets [**-regexp**] [**-nocase**] [**-filter** *arg*] [**-quiet**]
[**-verbose**] [*patterns*]

Returns

List of fileset objects

Usage

Name	Description
[**-regexp**]	Patterns are full regular expressions
[**-nocase**]	Perform case-insensitive matching (valid only when -regexp specified)
[**-filter**]	Filter list with expression
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*patterns*]	Match fileset names against patterns Default: *

Categories

Object, Project

Description

Gets a list of filesets in the current project that match a specified search pattern. The default command gets 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 (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

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

-filter *args* - (Optional) 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 the properties on an object with the **report_property** or **list_property** commands. In the case of the fileset object, "DESIGN_MODE", and "FILESET_TYPE" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

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

Examples

The following example returns the source files in the Source Set:

```
get_files -of_objects [get_filesets sources_1]
```

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*, you will get a warning that no filesets matched the specified pattern. However, in the above example the use of **-quiet** will suppress that warning message.

The following example gets 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

- get_files
- report_property

get_hierarchy_separator

Get hierarchical separator character.

Syntax

```
get_hierarchy_separator [-quiet] [-verbose]
```

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

SDC, XDC

Description

Gets the character currently used for separating levels of hierarchy in the design. You can set the hierarchy separator using the **set_hierarchy_separator** command.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example gets 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] [-filter arg]
[-of_objects args] [-quiet] [-verbose] [patterns]
```

Returns

List of interface objects

Usage

Name	Description
[**-regexp**]	Patterns are full regular expressions
[**-nocase**]	Perform case-insensitive matching (valid only when -regexp specified)
[**-filter**]	Filter list with expression
[**-of_objects**]	Get interfaces of these pins or nets
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*patterns*]	Match I/O port interfaces against patterns Default: *

Categories

Object

Description

Gets a list of IO interfaces in the current project that match a specified search pattern. The default command gets 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 (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

-nocase - (Optional) 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_interfaces** based on property values on the interfaces. You can find the properties on an object with the **report_property** or **list_property** commands.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects *args* - One or more pins or nets to which the interfaces are assigned.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - Match interfaces against the specified pattern. The default pattern is the wildcard '*' which gets a list of all interfaces in the project.

Examples

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

```
get_interfaces
```

See Also

- create_interface
- delete_interface

get_io_standards

Get a list of IO standards.

Syntax

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

Returns

IO standards

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get the IO standards of these bels, sites, package_pins, io_banks, ports.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[*patterns*]	Match IO standards against patterns Default: *

Categories

Object

get_iobanks

Get a list of iobanks.

Syntax

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

Returns

Iobanks

Usage

Name	Description
[**-regexp**]	Patterns are full regular expressions
[**-nocase**]	Perform case-insensitive matching. (valid only when -regexp specified)
[**-filter**]	Filter list with expression
[**-of_objects**]	Get the iobanks of these package_pins.
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*patterns*]	Match iobanks against patterns Default: *

Categories

XDC, Object

Description

Gets a list of I/O Banks on the target device in the current project that match a specified search pattern. The default command gets 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 (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

-nocase - (Optional) 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 the properties on an object with the **report_property** or **list_property** commands. Some of the properties that can be used with for an iobank object include "DCI_CASCADE", and "INTERNAL_VREF".

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects *arg* - Get a list of the I/O Banks connected to these objects. Valid object types are package_pins, ports, and sites.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - Match I/O Banks against the specified pattern. The default pattern is the wildcard '*' which gets a list of all I/O Banks in the design.

Examples

The following example returns the I/O Bank of the specified package pin:

```
get_iobanks -of_objects [get_package_pins H4]
```

See Also

- get_package_pins
- get_ports
- get_sites
- report_property

get_ipdefs

Get a list of IP from the current IP Catalog.

Syntax

get_ipdefs [**-regexp**] [**-nocase**] [**-filter** *arg*] [**-quiet**] [**-verbose**]
[*patterns*...]

Returns

List of Catalog IP objects

Usage

Name	Description
[**-regexp**]	Patterns are full regular expressions
[**-nocase**]	Perform case-insensitive matching
[**-filter**]	Filter list with expression
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*patterns*]	Match Catalog IP names against patterns Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

Object, IPFlow

Description

Get a list of IP cores defined in the IP catalog of the current project, based on the specified search pattern. The default is to return all IP cores defined in the catalog.

Arguments

-regexp - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

-nocase - (Optional) 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_ipdefs** based on property values on the objects. You can find the properties on an object with the **report_property** or **list_property** commands. In the case of the "ipdefs" object, "VLNV", "NAME" and "IS_AXI" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - Match IP core definitions in the IP catalog against the specified search patterns. The default pattern is the wildcard '*' which gets a list of all IP cores in the catalog. More than one pattern can be specified to find multiple core definitions 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 all IP cores with NAME property matching the specified pattern:

```
get_ipdefs -filter {NAME=~*agilent*}
```

Note The filter operator '=~' loosely matches the specified pattern

The following example returns a list of all AXI compliant IP cores:

```
get_ipdefs -filter {IS_AXI==1}
```

See Also

- create_ip
- generate_target
- get_ips
- import_ip
- update_ip_catalog

get_ips

Get a list of IPs in the current design.

Syntax

```
get_ips [-regexp] [-nocase] [-filter arg] [-quiet] [-verbose]
[patterns...]
```

Returns

List of IP objects

Usage

Name	Description
[**-regexp**]	Patterns are full regular expressions
[**-nocase**]	Perform case-insensitive matching
[**-filter**]	Filter list with expression
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*patterns*]	Match IP names against patterns Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

Object, Project, IPFlow

Description

Get a list of IP cores in the current project based on the specified search pattern. The default command returns a list of all IPs in the project.

Note To improve memory and performance, the get_* commands return a container list of a single type of objects (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

-nocase - (Optional) 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_ips** based on property values on the objects. You can find the properties on an object with the **report_property** or **list_property** commands. In the case of the "IP" object, "NAME" and "DELIVERED_TARGETS" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - Match IP cores in the design against the specified search patterns. The default pattern is the wildcard '*' which gets a list of all IP cores in the project. More than one pattern can be specified to find multiple cores 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 IP cores with names beginning with the string "EDK":

```
get_ips EDK*
```

See Also

- create_ip
- generate_target
- get_ipdefs
- import_ip
- update_ip_catalog

get_lib_cells

Get a list of Library Cells.

Syntax

```
get_lib_cells [-regexp] [-filter arg] [-nocase]
[-include_unsupported] [-of_objects args] [-quiet]
[-verbose] patterns
```

Returns

List of library cells

Usage

Name	Description
[-regexp]	Patterns are regular expressions
[-filter]	Filter list with expression
[-nocase]	Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.
[-include_unsupported]	Include test-only library cells.
[-of_objects]	Get the library cells of the objects passed in here: .
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
patterns	Match library cell names against patterns.

Categories

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 (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

-nocase - (Optional) 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_cells** based on property values on the cells. You can find the properties on an object with the **report_property** or **list_property** commands.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects *arg* - Get a list of library cells of specific instances (cells/insts), or library pins (get_lib_pins).

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - Match library cells against the specified patterns. The pattern must specify both the library name and the cell name.

Examples

The following example gets the number of the cells in the library for the target part in the current design, and then gets 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 gets 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

Get a list of Library Cell Pins.

Syntax

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

Returns

List of library cell pins

Usage

Name	Description
[-regexp]	Patterns are regular expressions
[-filter]	Filter list with expression
[-nocase]	Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.
[-of_objects]	Get the library cell pins of the objects passed in here.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
patterns	Match library cell pin names against patterns of the form /.

Categories

Object

Description

Gets 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 (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

-nocase - (Optional) 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 the properties on an object with the **report_property** or **list_property** commands.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects *arg* - Get a list of library cell pins of the specified pin objects or library cells (get_lib_cells).

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - Match lib pins against the specified patterns. The pattern must specify the library name, cell name, and the pins.

Examples

The following example gets a list of all library cell pins:

```
get_lib_pins xt_virtex6/AND2/*
```

The following example gets a list of 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

Get a list of Libraries.

Syntax

get_libs [**-regexp**] [**-filter** *arg*] [**-nocase**] [**-quiet**] [**-verbose**] [*patterns*]

Returns

List of libraries

Usage

Name	Description
[**-regexp**]	Patterns are regular expressions
[**-filter**]	Filter list with expression
[**-nocase**]	Perform case-insensitive matching when a pattern has been specified. This argument applies to the use of -regexp only.
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*patterns*]	Match library names against patterns. Default: *

Categories

Object

Description

Gets 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 (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

-nocase - (Optional) 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 the properties on an object with the **report_property** or **list_property** commands.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects *arg* - Get a list of libraries of the specified object.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - Match libraries against the specified patterns. The default pattern is the wildcard '*' which gets a list of all libraries in the project.

Examples

The following example gets 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**] [**-verbose**]

Returns

Message count

Usage

Name	Description
[**-severity**]	Message severity to query (not valid with -id,) e.g. "ERROR" or "CRITICAL WARNING" Default: ALL
[**-id**]	Unique message id to be queried (not valid with -severity,) e.g "Common 17-99"
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Report

Description

Gets the number of messages, of a specific severity or message ID, that have been returned by the tool since it was invoked.

Every message delivered by the tool has a unique global message ID that consists of an application sub-system code and a message identifier. This results in a message ID that looks like the following:

```
"Common 17-54"
"Netlist 29-28"
"Synth 8-3295"
```

This command can give you an idea of how close to the message limit the 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

-severity *value* - (Optional) 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 {} or "".

- **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.

-id *value* - (Optional) The message ID is found in the tool in the Messages view or other reports when the message is reported. Use the specific message ID to get the count for that message.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example gets 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 gets the message count of the specified message ID:

```
get_msg_count -id "Netlist 29-28"
```

See Also

- get_msg_limit
- set_msg_limit

get_msg_limit

Get message limit.

Syntax

get_msg_limit [**-severity** *arg*] [**-id** *arg*] [**-quiet**] [**-verbose**]

Returns

Message limit

Usage

Name	Description
[**-severity**]	Message severity to query (not valid with -id,) e.g. "ERROR" or "CRITICAL WARNING" Default: ALL
[**-id**]	Unique message id to be queried (not valid with -severity,) e.g "Common 17-99"
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Report

Description

Gets the number of messages that will be reported by the tool while invoked. When the tool 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.

Every message delivered by the tool has a unique global message ID that consists of an application sub-system code and a message identifier. This results in a message ID that looks like the following:

```
"Common 17-54"
"Netlist 29-28"
"Synth 8-3295"
```

Arguments

-id *arg* - The message ID to return the limit of. For example, "Common 17-54" or "Netlist 29-28".

-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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

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 29-28"
```

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]
[-top_net_of_hierarchical_group] [-segments]
[-boundary_type arg] [-quiet] [-verbose] [patterns]
```

Returns

List of net objects

Usage

Name	Description
[-hsc]	Hierarchy separator Default: /
[-hierarchical]	Search level-by-level in current instance
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get nets of these pins/ports,cells,timing paths or clocks
[-match_style]	Style of pattern matching, valid values are ucf, sdc Default: sdc
[-top_net_of_hierarchical _group]	Return net segment(s) which belong(s) to the high level of a hierarchical net
[-segments]	Return all segments of a net across the hierarchy
[-boundary_type]	Return net segment connected to a hierarchical pin which resides at the same level as the pin (upper) or at the level below (lower), or both. Valid values are : upper, lower, both Default: upper
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[*patterns*]	Match net names against patterns Default: *

Categories

SDC, XDC, Object

Description

Gets a list of nets in the current design that match a specified search pattern. The default command gets 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 (e.g. cells, nets, pins, or 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* - (Optional) The default hierarchy separator is '/'. Use this argument to specify a different hierarchy separator.

-hierarchical - (Optional) Get nets from all levels of the design hierarchy. Without this argument, the command will only get nets from the top of the design hierarchy. When using **-hierarchical**, the search pattern should not contain a hierarchy separator because the search pattern is applied at each level of the hierarchy, not to the full hierarchical cell name. For instance, searching for U1/* searches each level of the hierarchy for instances with U1/ in the name. This may not return the intended results. See **get_cells** for examples of **-hierarchical** searches.

Note When used with **-regexpr**, the specified search string is matched against the full hierarchical name, and the U1/* search pattern will work as intended

-regexp - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

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

-filter *args* - (Optional) 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 the properties 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 filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects *arg* - (Optional) Get a list of the nets connected to the specified cell, pin, port, or clock.

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

-top_net_of_hierarchical_group - (Optional) Get the top 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 - (Optional) Get all the segments of a hierarchical net, across all levels of the hierarchy. This differs from the **-hierarchical** argument in that it returns all segments of the specified net, rather than just the specified net.

-boundary_type - (Optional) Gets 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. Valid values are upper, lower, or both. The default value is upper.

Note This argument must be used with the **-of_objects** argument to specify the hierarchical pin.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - (Optional) Match nets against the specified patterns. The default pattern is the wildcard '*' which returns a list of 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 gets 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 you will get a warning.

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_nodes

Get a list of nodes in the device.

Syntax

```
get_nodes [-of_objects args] [-regexp] [-nocase] [-filter arg]
[-uphill] [-downhill] [-flyover] [-from args] [-to args]
[-quiet] [-verbose] [patterns]
```

Returns

Nodes

Usage

Name	Description
[-of_objects]	Get 'node' objects of these types: 'net tile node site_pin wire'.
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-uphill]	Get the nodes uphill (driver) from the provided site_pin, node or in the given tile(s).
[-downhill]	Get the nodes downhill (loads) from the provided site_pin, node or in the given tile(s).
[-flyover]	Get the nodes that fly over the given tile(s).
[-from]	-from Return the nodes beginning at this pip or site pin. May be used in combination with uphill. Default is downhill. -all is implied.
[-to]	-to Return the nodes ending at this wire or site pin. May be used in combination with uphill. Default is downhill. -all is implied.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[patterns]	Match the 'node' objects against patterns. Default: *

Categories

Object

Description

Returns a list of nodes on the device that match a specified search pattern in an open design.

The default command gets a list of all nodes on the device.

Note To improve memory and performance, the get_* commands return a container list of a single type of objects (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

-nocase - (Optional) 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_nodes** based on property values on the nodes. You can find the properties 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 node object, "IS_INPUT_PIN", "IS_BEL_PIN" and "NUM_WIRES" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects *args* - Return the nodes of the specified site_pins, nodes, tiles, or wires.

-uphill - Return the nodes uphill of the specified site_pin, node, or tile. Uphill nodes precede the specified object in the logic network.

-downhill - Return the nodes downhill of the specified site_pin, node, or tile. Downhill nodes follow the specified object in the logic network.

-flyover - Return the nodes that pass through (or flyover) the specified tiles.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - Return nodes matching the specified search patterns. The default pattern is the wildcard '*' which gets a list of all nodes on the device. More than one search pattern can be specified to find nodes 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 nodes associated with the specified tile:

```
get_nodes -of_objects [get_tiles CLBLM_R_X11Y158]
```

The following example returns the nodes downhill from the specified node:

```
get_nodes -downhill LIOB33_SING_X0Y199/IOB_PADOUT0
```

See Also

- get_nodes
- get_site_pins
- get_tiles
- get_wires
- list_property
- report_property

get_package_pins

Get a list of package pins.

Syntax

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

Returns

List of package pin objects

Usage

Name	Description
[**-regexp**]	Patterns are full regular expressions
[**-nocase**]	Perform case-insensitive matching. (valid only when -regexp specified)
[**-filter**]	Filter list with expression
[**-of_objects**]	Get the list of package pin objects of these sites iobanks ports.
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*patterns*]	Match list of package pin objects against patterns Default: *

Categories

XDC, Object

Description

Gets a list of the pins on the selected package for the target device. The default command gets 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 (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

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

-filter *args* - (Optional) 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 pins. You can find the properties on an object with the **report_property** or **list_property** commands. In the case of the package pin object, "IS_CLK_CAPABLE", "IS_VREF" and "IS_GLOBAL_CLK" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects *arg* - (Optional) Get the package pins connected to the specified objects. Valid objects include sites, I/O Banks, or ports.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - (Optional) 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 gets a list of all pins on the package of the target device:

```
get_package_pins
```

The following example gets 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 you will get a warning.

See Also

- get_iobanks
- get_sites
- list_property
- report_property

get_param

Get a parameter value.

Syntax

get_param [**-quiet**] [**-verbose**] *name*

Returns

Parameter value

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
name	Parameter name

Categories

PropertyAndParameter

Description

This command gets the currently defined value for a specified 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 - 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example gets 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] [-verbose]
[patterns]
```

Returns

List of part objects

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[patterns]	Match part names against patterns Default: * Values: The default search pattern is the wildcard *, or .* when -regexp is specified.

Categories

Object

Description

Gets a list of parts in the current project that match a specified search pattern. The default command gets 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 (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

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

-filter *args* - (Optional) 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 the properties 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 part object, "DEVICE", "FAMILY" and "SPEED" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - (Optional) Match parts against the specified patterns. The default pattern is the wildcard '*' which gets a list of 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 gets a list of 7vx485t parts, speed grade -1:

```
get_parts -filter {DEVICE =~ xc7vx485t* && speed == -1}
```

The following example gets 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 tool will return a warning.

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] [-verbose]
[patterns]
```

Returns

List of path groups

Usage

Name	Description
[**-regexp**]	Patterns are full regular expressions
[**-nocase**]	Perform case-insensitive matching (valid only when -regexp specified)
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*patterns*]	Match path group names against patterns Default: *

Categories

XDC, Object

Description

Gets a list of timing path groups in the current project that match a specified search pattern. The default command gets a list of all path groups in the design.

Path groups are automatically created when a new clock is created in the design, containing all paths in that clocks domain. Path groups can also be manually created with the use of the **group_path** command.

Note To improve memory and performance, the get_* commands return a container list of a single type of objects (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

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

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - Match path groups against the specified patterns. The default pattern is the wildcard '*' which gets all path groups in the project.

Examples

The following example gets a list of all the path groups in the design.

```
get_path_groups
```

The following example gets all path groups with the string "Clk" somewhere in the name:

```
get_path_groups *Clk*
```

Note If no path groups match the pattern you will get a warning.

get_pblocks

Get a list of pblocks in the current design.

Syntax

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

Returns

List of pblock objects

Usage

Name	Description
[**-regexp**]	Patterns are full regular expressions
[**-nocase**]	Perform case-insensitive matching (valid only when -regexp specified)
[**-filter**]	Filter list with expression
[**-of_objects**]	Get pblocks of these cells
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*patterns*]	Match pblock names against patterns Default: *

Categories

Object, Floorplan, XDC

Description

Gets a list of Pblocks defined in the current project that match a specific pattern. The default command gets 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 (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

-nocase - (Optional) 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_pblocks** based on property values on the Pblocks. You can find the properties on an object with the **report_property** or **list_property** commands. In the case of the Pblock object, "NAME" and "GRID_RANGES" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects *arg* - Get the Pblocks to which the specified cells are assigned.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

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

Examples

The following example gets a list of all Pblocks in the current project:

```
get_pblocks
```

The following example gets a list of all Pblocks which do not have a Slice Range defined:

```
get_pblocks -filter {GRIDTYPES !~ SLICE}
```

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]
[-verbose] [patterns]
```

Returns

List of pin objects

Usage

Name	Description
[**-hsc**]	Hierarchy separator Default: /
[**-hierarchical**]	Search level-by-level in current instance
[**-regexp**]	Patterns are full regular expressions
[**-nocase**]	Perform case-insensitive matching (valid only when -regexp specified)
[**-leaf**]	Get leaf/global pins of nets with -of_objects
[**-filter**]	Filter list with expression
[**-of_objects**]	Get pins of these cells, nets, bel pins, timing paths or clocks
[**-match_style**]	Style of pattern matching, valid values are ucf, sdc Default: sdc
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*patterns*]	Match pin names against patterns Default: *

Categories

SDC, XDC, Object

Description

Gets a list of pin objects in the current design that match a specified search pattern. The default command gets 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 (e.g. cells, nets, pins, or 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* - (Optional) The default hierarchy separator is '/'. Use this argument to specify a different hierarchy separator.

-hierarchical - (Optional) Get pins from all levels of the design hierarchy. Without this argument, the command will only get pins from the top of the design hierarchy. When using **-hierarchical**, the search pattern should not contain a hierarchy separator because the search pattern is applied at each level of the hierarchy, not to the full hierarchical cell name. For instance, searching for U1/* searches each level of the hierarchy for instances with U1/ in the name. This may not return the intended results. See **get_cells** for examples of **-hierarchical** searches.

Note When used with **-regexpr**, the specified search string is matched against the full hierarchical name, and the U1/* search pattern will work as intended

-regexp - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

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

-leaf - (Optional) Include leaf pins, from primitive or black box cells, for the objects specified with the **-of_object** argument.

-filter *args* - (Optional) 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 the properties 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 filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects *arg* - (Optional) Get the pins connected to the specified cell, port, or clock.

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

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - (Optional) Match pins against the specified patterns. The default pattern is the wildcard '*' which gets a list of 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.

Examples

The following example gets 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 tool will return a warning.

See Also

- list_property
- report_property

get_pips

Get a list of programmable interconnect points (pips) on the current device.

Syntax

```
get_pips [-regexp] [-nocase] [-filter arg] [-of_objects args]
[-uphill] [-downhill] [-from args] [-to args] [-quiet]
[-verbose] [patterns]
```

Returns

Pips

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get the pips of these sites, tiles, wires, nodes, pips, or nets.
[-uphill]	Get the pips uphill from the provided wire or pip.
[-downhill]	Get the pips downhill from the provided wire or pip.
[-from]	-from Return the ordered list of pips beginning at this pip or site pin. May be used in combination with uphill. Default is downhill. -all is implied.
[-to]	-to Return the ordered list of pips ending at this wire or site pin. May be used in combination with uphill. Default is downhill. -all is implied.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[*patterns*]	Match pips against patterns Default: *

Categories

Object

Description

Programmable interconnect points, or PIPs, provide the physical routing paths on the device used to connect logic networks. This command returns a list of PIPs on the device that match a specified search pattern. The command requires a design to be open.

The default command gets a list of all PIPs on the device. However, this is not a recommended use of the command due to the number of pips on a device. You should specify the **-of_objects** argument to limit the number of pips returned.

Note To improve memory and performance, the get_* commands return a container list of a single type of objects (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

-nocase - (Optional) 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_pips** based on property values on the PIPs. You can find the properties 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 PIP object, "IS_DIRECTIONAL" and "FROM_PIN" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects *args* - Return the PIPs of the specified site, tile, or wire objects.

Note Xilinx recommends that you always use the **-of_objects** argument to limit the runtime and memory used by the **get_pips** command. The number of programmable interconnect points returned can be considerable

-uphill - Return the PIPs uphill of the specified wire or PIPs. Uphill PIPs precede the specified object in the logic network.

-downhill - Return the PIPs downhill of the specified wire or PIPs. Downhill PIPs follow the specified object in the logic network.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - Return PIPs matching the specified search patterns. The default pattern is the wildcard '*' which gets a list of all PIPs on the device. More than one search pattern can be specified to find PIPs 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 PIPs associated with the specified tile:

```
get_pips -of_object [get_tiles DSP_R_X9Y75]
```

See Also

- get_sites
- get_tiles
- get_wires
- 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] [-verbose] [patterns]
```

Returns

List of port objects

Usage

Name	Description
[**-regexp**]	Patterns are full regular expressions
[**-nocase**]	Perform case-insensitive matching (valid only when -regexp specified)
[**-filter**]	Filter list with expression
[**-of_objects**]	Get ports of these nets, instances, sites, clocks, timing paths, io standards, io banks, package pins
[**-match_style**]	Style of pattern matching, valid values are ucf, sdc Default: sdc
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*patterns*]	Match port names against patterns Default: *

Categories

SDC, XDC, Object

Description

Gets a list of port objects in the current design that match a specified search pattern. The default command gets 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 (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

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

-filter *args* - (Optional) 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 the properties 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 filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects *arg* - Get the ports connected to the specified cell, net, clock, or timing path objects.

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

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - Match ports against the specified patterns. The default pattern is the wildcard '*' which gets a list of 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.

Examples

The following example gets 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 tool will return a warning.

See Also

- list_property
- report_property

get_projects

Get a list of projects.

Syntax

```
get_projects [-regexp] [-nocase] [-filter arg] [-quiet]
[-verbose] [patterns]
```

Returns

List of project objects

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching (valid only when -regexp specified)
[-filter]	Filter list with expression
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[patterns]	Match project names against patterns Default: *

Categories

Object, Project

Description

Gets a list of open projects that match the specified search pattern. The default gets 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 (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

-nocase - (Optional) 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_projects** based on property values on the projects. You can find the properties 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 filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

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

Examples

The following example gets 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 you will get a warning.

See Also

- create_project
- current_project
- open_project

get_property

Get properties of object.

Syntax

get_property [**-quiet**] [**-verbose**] *name object*

Returns

Property value

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
name	Name of property whose value is to be retrieved
object	Object to query for properties

Categories

Object, PropertyAndParameter

Description

Gets 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

name - The name of the property to be returned. The name is not case sensitive.

object - The object to query.

Examples

The following example gets 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] [-verbose] [patterns]
```

Returns

List of reconfigurable module objects

Usage

Name	Description
[**-regexp**]	Patterns are full regular expressions
[**-nocase**]	Perform case-insensitive matching (valid only when -regexp specified)
[**-filter**]	Filter list with expression
[**-of_objects**]	Get Reconfigurable Modules for these cells
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*patterns*]	Match Reconfigurable Module names against patterns

Categories

Object, PartialReconfiguration

Description

Gets a list of reconfigurable modules in the current project that match a specified search pattern. The default command gets 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 (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

-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 the properties on an object with the **report_property** or **list_property** commands.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_object *arg* - Get reconfigurable modules from the specified object.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - Match reconfigurable modules against the specified patterns. The default pattern is the wildcard '*' which gets a list of all modules in the project.

Examples

The following example gets 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] [-verbose]
[patterns]
```

Returns

List of run objects

Usage

Name	Description
[**-regexp**]	Patterns are full regular expressions
[**-nocase**]	Perform case-insensitive matching (valid only when -regexp specified)
[**-filter**]	Filter list with expression
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*patterns*]	Match run names against patterns Default: *

Categories

Object, Project

Description

Gets a list of synthesis and implementation runs in the current project that match a specified search pattern. The default command gets 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 (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

-nocase - (Optional) 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 the properties 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 runs object, "CONSTRSET", "IS_IMPLEMENTATION", "IS_SYNTHESIS", and "FLOW" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - Match run names against the specified patterns. The default pattern is the wildcard '*' which gets a list of 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 gets a list of all incomplete runs in the current project:

```
get_runs -filter {PROGRESS < 100}
```

The following example gets a list of runs matching the specified pattern:

```
get_runs imp*
```

Note If there are no runs matching the pattern you will get a warning.

See Also

report_property

get_selected_objects

Get selected objects.

Syntax

```
get_selected_objects [-primary] [-quiet] [-verbose]
```

Returns

List of selected objects

Usage

Name	Description
[**-primary**]	Do not include objects that were selected due to selection rules
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Object, GUIControl

Description

Gets the objects currently selected by the **select_objects** command. Can get the primary selected object and any secondary selected objects as well.

Primary objects are directly selected, while secondary objects are selected 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

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_site_pins

Get a list of site_pins.

Syntax

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

Returns

Site_pins

Usage

Name	Description
[**-of_objects**]	Get 'site_pin' objects of these types: 'site xdef_site node pin'.
[**-regexp**]	Patterns are full regular expressions
[**-nocase**]	Perform case-insensitive matching. (valid only when -regexp specified)
[**-filter**]	Filter list with expression
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*patterns*]	Match the 'site_pin' objects against patterns. Default: *

Categories

Object

Description

Returns a list of site pins of the specified site, BELs, or logical cell pin objects in an open design. This command requires the use of the **-of_objects** argument.

Note To improve memory and performance, the get_* commands return a container list of a single type of objects (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

-nocase - (Optional) 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_site_pins** based on property values on the site pins. You can find the properties 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 site pin object, "IS_CLOCK", "IS_DATA" and "IS_PART_OF_BUS" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects *args* - Return the site pins of specified site, routed sites, BELs, or logical cell pin objects.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example returns the site_pins of the specified BELs:

```
get_site_pins -of_objects [get_bels SLICE_X21Y92/B5FF]
CE CK D SR Q
get_site_pins -of_objects [get_bels SLICE_X21Y92/A5LUT]
A1 A2 A3 A4 A5 O5
get_site_pins -of_objects [get_bels SLICE_X21Y92/CARRY4_CMUX]
0 1 S0 OUT
```

The following example returns the site_pins associated with the specified site:

```
get_site_pins -of_objects [get_sites SLICE_X21Y92]
A1 A2 A3 A4 A5 A6 AX B1 B2 B3 B4 B5 B6 BX C1 C2 C3 C4 C5 C6 CE CIN CLK CX \
D1 D2 D3 D4 D5 D6 DX SR A AMUX AQ B BMUX BQ C CMUX COUT CQ D DMUX DQ
```

See Also

- get_bels
- get_sites
- list_property
- report_property

get_site_pips

Get a list of site_pips from the given object.

Syntax

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

Returns

Site_pips

Usage

Name	Description
[-regexp]	Patterns are full regular expressions
[-nocase]	Perform case-insensitive matching. (valid only when -regexp specified)
[-filter]	Filter list with expression
[-of_objects]	Get the site_pips of these sites.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[*patterns*]	Match site_pips against patterns Default: *

Categories

Object

Description

Programmable interconnect points, or PIPs, provide the physical routing paths on the device used to connect logic networks. This command returns a list of PIPs on specified sites that match a specified search pattern. The command requires a design to be open.

This command requires the use of the -of_objects option to specify the sites to return PIPs from.

Note To improve memory and performance, the get_* commands return a container list of a single type of objects (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

-nocase - (Optional) 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_site_pips** based on property values on the PIPs. You can find the properties 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 PIP object, "IS_DIRECTIONAL" and "FROM_PIN" are two of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects *args* - This option can be used with the **get_bels** command to return the pins of specified BELs.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - Match BEL pins against the specified patterns. The default pattern is the wildcard '*' which gets a list of all BEL pins on the device. More than one search pattern can be specified to find pins 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 pins of the specified BELs associated with the specified range of sites on the device:

```
get_site_pips -of_objects [get_sites SLICE_X21Y92]
```

The following example returns the clock enable (CE) pins of of all BELs on the device:

```
get_bel_pins *CE
```

See Also

- get_bels
- get_sites
- list_property
- report_property

get_sites

Get a list of Sites.

Syntax

```
get_sites [-regexp] [-filter arg] [-range args]
[-of_objects args] [-quiet] [-verbose] [patterns]
```

Returns

List of site objects

Usage

Name	Description
[**-regexp**]	Patterns are full regular expressions
[**-filter**]	Filter list with expression
[**-range**]	Match site names which fall into the range. Range is defined by exactly two site names.
[**-of_objects**]	Get the sites of the objects passed in here.
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*patterns*]	Match site names against patterns. Bonded sites will also match on package pin names. Default: *

Categories

XDC, Object

Description

Gets a list of sites on the target device that match a specified search pattern. The default command gets 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 (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

-nocase - (Optional) 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_sites based on property values on the sites. You can find the properties on an object with the **report_property** or **list_property** commands. 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.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-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. Valid objects include: tiles, BELs, pins, package pins, ports, Pblocks, I/O Banks, cells, and clock_regions.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - Match sites against the specified patterns. The default pattern is the wildcard '*' which gets a list of all sites on the target device.

Examples

The following example gets a list of all sites available on the target device:

```
get_sites
```

The following example gets the number of unoccupied sites on the device:

```
llength [get_sites -filter {IS_OCCUPIED==0}]
```

Note If no sites match the pattern you will get a warning.

The following example gets all of the sites on the device, and returns the unique SITE_TYPEs:

```
set sites [get_sites]
set type {}
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 shows three different forms for specifying the range of sites to return:

```
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
get_sites -range {SLICE_X0Y0:SLICE_X1Y1}
SLICE_X0Y0 SLICE_X0Y1 SLICE_X1Y0 SLICE_X1Y1
```

See Also

- get_cells
- get_property
- list_property
- report_property

get_slrs

Get a list of Super Logic Regions (SLRs).

Syntax

get_slrs [**-regexp**] [**-filter** *arg*] [**-no_case**] [**-of_objects** *args*]
[**-quiet**] [**-verbose**] [*patterns*]

Returns

List of SLRs

Usage

Name	Description
[**-regexp**]	Patterns are full regular expressions
[**-filter**]	Filter list with expression
[**-no_case**]	Perform case-insensitive matching. (valid only when -regexp specified)
[**-of_objects**]	Get the SLRs of the objects passed in here.
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*patterns*]	Match SLR names against patterns. Default: *

Categories

Object

get_tiles

Get a list of tiles.

Syntax

get_tiles [**-regexp**] [**-nocase**] [**-filter** *arg*] [**-of_objects** *args*]
[**-quiet**] [**-verbose**] [*patterns*]

Returns

Tiles

Usage

Name	Description
[**-regexp**]	Patterns are full regular expressions
[**-nocase**]	Perform case-insensitive matching. (valid only when -regexp specified)
[**-filter**]	Filter list with expression
[**-of_objects**]	Get the tiles of these sites, bels, site_pins, nodes, wires, pips, nets, clock_regions.
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*patterns*]	Match tiles against patterns Default: *

Categories

Object

Description

This command returns a list of tiles on the device in an open design. The default command gets a list of all tiles on the device.

Note To improve memory and performance, the get_* commands return a container list of a single type of objects (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

-nocase - (Optional) 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_tiles** based on property values on the tile objects. You can find the properties 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 tile object, "NUM_ARCS", "NUM_SITES", and "IS_GT_SITE_TILE" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects *args* - Can be used to return the tiles associated with specified sites, BELs, site_pins, nodes, wires, and PIPs.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - Match tiles against the specified patterns. The default pattern is the wildcard '*' which gets a list of all tiles on the device. More than one search pattern can be specified to find tiles 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 total number of tiles where the number of timing arcs is greater than 100 and 150 respectively:

```
llength [get_tiles -filter {NUM_ARCS>100} ]
13468
llength [get_tiles -filter {NUM_ARCS>150} ]
11691
```

See Also

- get_bels
- get_nodes
- get_pips
- get_site_pins
- get_sites
- get_wires
- list_property
- report_property

get_wires

Get a list of wires.

Syntax

```
get_wires [-regexp] [-nocase] [-filter arg] [-of_objects args]
[-uphill] [-downhill] [-from args] [-to args] [-quiet]
[-verbose] [patterns]
```

Returns

Wires

Usage

Name	Description
[**-regexp**]	Patterns are full regular expressions
[**-nocase**]	Perform case-insensitive matching. (valid only when -regexp specified)
[**-filter**]	Filter list with expression
[**-of_objects**]	Get the wires of these tiles, nodes, pips, or nets.
[**-uphill**]	Get the wires uphill from the provided pip.
[**-downhill**]	Get the wires downhill from the provided pip.
[**-from**]	-from Return the ordered list of wires beginning at this pip or site pin. May be used in combination with uphill. Default is downhill. -all is implied.
[**-to**]	-to Return the ordered list of wires ending at this wire or site pin. May be used in combination with uphill. Default is downhill. -all is implied.
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*patterns*]	Match wires against patterns Default: *

Categories

Object

Description

Returns a list of wires in the design that match a specified search pattern in an open design.

The default command gets a list of all wires in the design.

Note To improve memory and performance, the get_* commands return a container list of a single type of objects (e.g. cells, nets, pins, or 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 - (Optional) Specifies that the search *patterns* are written as regular expressions. Both search *patterns* and **-filter** expressions must be written as regular expressions when this argument is used. Xilinx regular expression Tcl commands are always anchored to the start of the search string. You can add ".*" to the beginning or end of a search string to widen the search to include a substring. See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for help with regular expression syntax.

Note The Tcl built-in command **regexp** is not anchored, and works as a standard Tcl command. For more information refer to http://www.tcl.tk/man/tcl8.5/TclCmd/regexp.htm.

-nocase - (Optional) 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_wires** based on property values on the wires. You can find the properties 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 wire object, "NAME", "NUM_DOWNHILL_PIPS" and "NUM_UPHILL_PIPS" are some of the properties that can be used to filter results.

The filter search pattern should be quoted to avoid having to escape special characters that may be found in net, pin, or cell names, or other properties. String matching is case-sensitive and is always anchored to the start and to the end of the search string. The wildcard "*" character can be used at the beginning or at the end of a search string to widen the search to include a substring of the property value.

Note The filter returns an object if a specified property exists on the object, and the specified pattern matches the property value on the object. In the case of the "*" wildcard character, this will match a property with a defined value of "".

For string comparison, the specific operators that can be used in filter expressions are "equal" (==), "not-equal" (!=), "match" (=~), and "not-match" (!~). Numeric comparison operators <, >, <=, and >= can also be used. Multiple filter expressions can be joined by AND and OR (&& and ||). The following gets input ports that do NOT contain the "RESET" substring within their name:

```
get_ports * -filter {DIRECTION == IN && NAME !~ "*RESET*"}
```

Boolean (**bool**) type properties can be directly evaluated in filter expressions as true or not true:

```
-filter {IS_PRIMITIVE && !IS_LOC_FIXED}
```

-of_objects *args* - Return the wires of the specified nodes, PIPs, or tiles.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

patterns - Return wires matching the specified search patterns. The default pattern is the wildcard '*' which gets a list of all wires in the design. More than one search pattern can be specified to find wires 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 wires associated with the specified tile:

```
get_wires -of_objects [get_tiles IO_INT_INTERFACE_L_X0Y198]
```

See Also

- get_nodes
- get_pips
- get_tiles
- list_property
- report_property

help

Display help for one or more topics.

Syntax

help [**-category** *arg*] [**-args**] [**-syntax**] [**-long**] [**-quiet**]
[**-verbose**] [*pattern*]

Returns

Nothing

Usage

Name	Description
[**-category**]	Search for topics in the specified category
[**-args**]	Display arguments description
[**-syntax**]	Display syntax description
[**-long**]	Display long help description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*pattern*]	Display help for topics that match the specified pattern Default: *

Categories

Project

Description

Returns a brief or long description of the specified Tcl command; a list of available Xilinx Tcl command categories; or a list of commands matching a specific pattern.

The default **help** command without any arguments returns a list of Tcl command categories that can be further explored. Command categories are groups of commands performing a specific function, like File I/O commands for instance.

Different arguments for the **help** command can return just the command syntax for a quick reminder of how the command should be structured; the command syntax and a brief description of each argument; or the long form of the command with more detailed descriptions and examples of the command.

Arguments

-category *arg* - (Optional) Get a list of the commands grouped under the specified command category.

-syntax - (Optional) Returns only the syntax line for the command as a quick reminder.

-args - (Optional) Get abbreviated help text for the specified command. The default is to return the detailed help for the specified command. Use this argument to keep it brief.

-long - (Optional) Returns the extended help description for the command, including the syntax, a brief description of the arguments, and a more detailed description of the command with examples. This is the default setting for the help command.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

pattern - (Optional) Returns information related to the specified command, or a list of commands that match the specified pattern.

Examples

The following example returns a list of Xilinx Tcl command categories:

```
help
```

The following example returns a list of all commands matching 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 long description of the **remove_files** command and its arguments:

```
help remove_files
```

Note You can also use the **-args** option to get a brief description of the command.

The following example defines a procedure called short, and returns the **-args** form of help for the specified command:

```
proc short cmdName {help -args $cmdName}
```

Note You can add this procedure to the `$HOME/.Xilinx/PlanAhead/init.tcl` file to load this command every time the tool is launched. Refer to *Chapter 1, Introduction* for more information on the `init.tcl` file

highlight_objects

Highlight objects in different colors.

Syntax

```
highlight_objects [-color_index arg] [-rgb args] [-color arg]
[-quiet] [-verbose] objects
```

Returns

Nothing

Usage

Name	Description
[**-color_index**]	Color index
[**-rgb**]	RGB color index list
[**-color**]	Valid values are red green blue magenta yellow cyan and orange
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
objects	Objects to highlight

Categories

GUIControl

Description

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 - 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 - 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 - 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

objects - Specifies one or more objects to be highlighted.

Examples

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] [-verbose] [cores...]
```

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*cores*]	ChipScope debug core

Categories

ChipScope

Description

Implements 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 tool automatically adds an ICON controller core to contain and configure the ILA cores in the project.

The 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

cores - (Optional) One or more debug cores to implement. All debug cores will be implemented if no cores are specified.

Examples

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]
[-verbose] ncd
```

Returns

Nothing

Usage

Name	Description
[-**run**]	Import the results into this run
[-**twx**]	TWX file to import
[-**pcf**]	PCF file to import
[-**quiet**]	Ignore command errors
[-**verbose**]	Suspend message limits during command execution
ncd	Routed NCD file to import

Categories

Project

Description

Imports 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 tool.

Arguments

-run *arg* - (Optional) The name of the implementation run to be imported into.

Note 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* - (Optional) 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* - (Optional) 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

ncd - 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 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_demo.pcf C:/Data/ise/drp_des/drp_demo.ncd
```

Note If the **reset_run** command was not used on the **impl_1** run, the following error is 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] [-verbose] [files...]
```

Returns

A list of file objects that were imported

Usage

Name	Description
[-**fileset**]	Fileset name
[-**force**]	Overwrite files of the same name in project directory
[-**norecurse**]	Disables the default behavior of recursive directory searches
[-**flat**]	Import the files into a flat directory structure
[-**relative_to**]	Import the files with respect to the given relative directory
[-**quiet**]	Ignore command errors
[-**verbose**]	Suspend message limits during command execution
[*files*]	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* - (Optional) The fileset to which the specified source files should be added. If the specified fileset does not exist, the tool will return an error. If no fileset is specified the files will be added to the source fileset by default.

-force - (Optional) Overwrite files of the same name in the local project directory and in the fileset.

-norecurse - (Optional) Do not recurse through subdirectories of any specified directories. Without this argument the tool will also search through any subdirectories for additional source files that can be added to a project.

-flat - (Optional) Import all files into the imports folder without preserving their relative paths. By default the directory structure of files is preserved as they are imported into the design.

-relative_to *arg* - (Optional) 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

files - (Optional) One or more file names or directory names to be added to the specified fileset. If a directory name is specified, all valid source files found in the directory, and in subdirectories of the directory, will be added. If no files are specified, the tool imports files in the source set for the current project.

Note If the path is not specified as part of the file name, the current working directory is used, or the directory from which the 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 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 a `\level1` directory would be created 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*] [**-files** *args*] [**-file** *arg*] [**-name** *arg*]
[**-quiet**] [**-verbose**]

Returns

List of file objects that were added

Usage

Name	Description
[**-srcset**]	Source set name
[**-files**]	Names of the IP files to be imported
[**-file**]	Name of the IP file to be imported and renamed (cannot be used alongside the -files option)
[**-name**]	New name for the imported IP (only valid when used with -file option)
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Project, IPFlow

Description

Imports an existing XCO file as an IP source into the current project. An XCO file is a CORE Generator 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 XCO files are used to recreate the core in the current project.

Note The PlanAhead tool reads IP from the ChipScope tool. It does not support reading Vivado native IP (XCI) files

The **import_ip** command allows you to read existing IP files directly, and import them into the local project folders. Use the **read_ip** or **add_files** command to add IP files by reference into the current project.

Use the **create_ip** command to create new IP files from the current IP catalog.

Arguments

-file *arg* - The IP file to be imported into the current project. The IP must be in the form of an existing XCO file.

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

-name *arg* - The name to assign to the IP object as it is added to the current source fileset.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

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.xco -name IP_block1
```

See Also

- add_files
- create_ip
- generate_target
- read_ip

import_synplify

Imports the given Synplify project file.

Syntax

import_synplify [**-copy_sources**] [**-quiet**] [**-verbose**] *file*

Returns

List of files object that were imported from the Synplify file

Usage

Name	Description
[**-copy_sources**]	Copy all the sources from synplify project file into the created project
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
file	Name of the Synplify project file to be imported

Categories

Project

Description

Imports Synplify synthesis project files (`.prj`) into the current project, including the various source files used in the synthesis run.

Arguments

-copy_sources - (Optional) Copy Synplify project source files to the local project directory structure rather than referencing them from their current location. The default is to reference source files from their current location.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

file - The name of the Synplify project file from which to import the source files.

Examples

The following example creates a new project and imports the specified Synplify project file, copying the various source files from the Synplify project into the local project directories:

```
create_project syn_test C:/Data/FPGA_Design/syn_test
import_synplify -copy_sources C:/Data/syn_data.prj
```

See Also

create_project

import_xise

Import XISE project file settings into the created project.

Syntax

import_xise [**-copy_sources**] [**-quiet**] [**-verbose**] *file*

Returns

True

Usage

Name	Description
[**-copy_sources**]	Copy all ISE sources into the created project
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
file	Name of the XISE project file to be imported

Categories

Project

Description

Imports an ISE project file (XISE) into the current project. This allows ISE projects to be quickly migrated for synthesis, simulation, and implementation. All project source files, constraint files, simulation files, and run settings are imported from the ISE project and recreated in the current 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 - Copy source files in the ISE project to the local project directory structure rather than referencing them from their current location. The default is to reference source files from their current location.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

file - The name of the ISE project file (.XISE) to be imported into the current project.

Examples

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_des/first_use.xise
```

Note 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 [**-copy_sources**] [**-quiet**] [**-verbose**] *file*

Returns

List of files object that were imported from the XST file

Usage

Name	Description
[**-copy_sources**]	Copy all the sources from xst project file into the created project
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
file	Name of the XST project file to be imported

Categories

Project

Description

Imports XST synthesis project files into the current project, including the various source files used in the XST run.

Arguments

-copy_sources - (Optional) Copy XST project source files to the local project directory structure rather than referencing them from their current location. The default is to reference source files from their current location.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

file - The name of the XST project file from which to import the source files.

Examples

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.xst
```

See Also

create_project

infer_diff_pairs

Infer differential pairs, typically for ports just imported from a CSV, UCF, or XDC file.

Syntax

```
infer_diff_pairs [-file_type arg] [-quiet] [-verbose] [file...]
```

Returns

Nothing

Usage

Name	Description
[-**file_type**]	Input file type: 'csv', 'ucf', or 'xdc' Default: file type
[-**quiet**]	Ignore command errors
[-**verbose**]	Suspend message limits during command execution
[*file*]	Pin Planning CSV file, UCF file, or XDC file Default: file

Categories

FileIO

Description

The **infer_diff_pairs** command can be used in an I/O Pin Planning project, after importing the I/O pin information using the **read_csv**, or **read_ucf** command.

There are several attributes that identify differential pairs in the file: Signal Name, DiffPair Signal, DiffPair Type, and I/O Standard.

The tool will identify differential pairs using the following methods:

- Matching Diff Pair - This is a direct definition of the two signals which make up a differential pair. Two port entries, each have DiffPair Signal values linking to the Signal Name of the other, and have complementary DiffPair Type values, one N and one P. The tool checks to ensure that the other attributes such as I/O Standard are compatible when forming the diff pair.

- Unmatched Diff Pair - Two port entries, with complementary DiffPair Type values (one N, one P), but only one port has a DiffPair Signal linking to the other Signal Name. The tool will create the differential pair if all other attributes are compatible.

- Single Port Diff Pair - A single port entry with a differential I/O Standard, a DiffPair Type value, and a DiffPair Signal that does not otherwise appear in the CSV. The tool will create the opposite side of the differential pair (the N or P side), with all properties matching those on the original port.

- Inferred Diff Pair - Two ports entries, with Signal Names that imply the N and P side. The tool will infer a differential pair if all other attributes are compatible.

After reading the port definitions from a CSV, or UCF file, the tool will report that some differential pairs can be inferred from the data. You can run the **infer_diff_pairs** command to infer these differential pairs if you choose.

Arguments

-file_type [csv | ucf | xdc] - Specify the type of file to import when inferring differential pairs. The valid file types are CSV, UCF, and XDC. There is no default; the **-file_type** must be specified.

Note XDC files are supported in the Vivado tool only, and UCF files are supported in the PlanAhead tool only.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

file - The name of the file previously imported.

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

Examples

The following example imports the specified UCF file, and then infers differential pairs from the file:

```
read_ucf C:/Data/io_1.ucf
infer_diff_pairs C:/Data/io_1.xdc -file_type ucf
```

See Also

- read_csv
- read_ucf

launch_chipscope_analyzer

Launch ChipScope Analyzer tool for a run.

Syntax

launch_chipscope_analyzer [**-run** *arg*] [**-csproject** *arg*] [**-quiet**]
[**-verbose**]

Returns

Nothing

Usage

Name	Description
[**-run**]	Implemented run to launch ChipScope Analyzer with
[**-csproject**]	ChipScope project
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

ToolLaunch, ChipScope

Description

Launches 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* - 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* - 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

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
```

Note 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

launch_fpga_editor

Launch FPGAEditor tool for a Run.

Syntax

launch_fpga_editor [**-run** *arg*] [**-more_options** *arg*] [**-mapped_ncd**]
[**-quiet**] [**-verbose**]

Returns

Nothing

Usage

Name	Description
[**-run**]	Implemented run to launch FED with
[**-more_options**]	More command line options
[**-mapped_ncd**]	Use Mapped NCD
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

ToolLaunch

Description

Launches 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 tool and the
FPGA Editor using the **crossprobe_fed** command to quickly locate elements of your
design in the two editors.

Arguments

-run *arg* - (Optional) 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* - (Optional) Additional options to use when invoking the FPGA
Editor software. For more information, see the online Help provided with FPGA Editor.

-mapped_ncd - (Optional) 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

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**] [**-verbose**]

Returns

Nothing

Usage

Name	Description
[**-run**]	Implemented run to launch iMPACT with
[**-ipf**]	Project for iMPACT
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

ToolLaunch

Description

Launch iMPACT to configure your device and generate programming files. You can also read back and verify design configuration data, debug configuration problems, or execute XSVF files.

You must generate the bitstream file using **write_bitstream** prior to using iMPACT.

The command returns the list of files read.

Arguments

-run - (Optional) Launch iMPACT with the specified run. If no run is specified, then iMPACT is launched with the active implementation run.

-ipf - (Optional) Specify the iMPACT project file to use to save the results to. The iMPACT Project File (IPF) contains information from a previous session of iMPACT. The target device is configured according to the settings in the specified IPF file. If you do not specify **-ipf**, the target device is configured according to the default settings.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example launches iMPACT using the specified implementation run:

```
launch_impact -run impl_3
```

launch_isim

Launch simulation using ISim simulator.

Syntax

launch_isim [**-simset** *arg*] [**-mode** *arg*] [**-noclean_dir**] [**-quiet**]
[**-verbose**]

Returns

Nothing

Usage

Name	Description
[**-simset**]	Name of the simulation fileset
[**-mode**]	Simulation mode. Values: behavioral, timing, post-implementation Default: behavioral
[**-noclean_dir**]	Do not remove simulation run directory files
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

ToolLaunch, Simulation

Description

Launch the Xilinx ISim simulator using the simulation source files in a simulation fileset. You can create a simulation fileset within the current project using the **create_fileset** command.

Launching the ISim simulator first runs fuse, the RTL elaborator, compiler, and linker used to create a simulation executable used by the simulator.

ISim is then launched, using the created executable.

The command returns messages related to running fuse and ISim.

Arguments

-simset - (Optional) Name of the simulation fileset containing the simulation test benches and sources to be used during simulation. If not specified, the current simulation fileset is used.

-noclean_dir - (Optional) Do not remove simulation run directory files prior to launching the simulator. However, some of the files generated for use by the simulator will be overwritten or updated by re-launching the simulator. The default is to remove the simulation run directory before launching the simulator.

-mode [behavioral | timing] - (Optional) Simulation mode. Specifies either a behavioral simulation of the design to verify syntax and confirm that the design performs as intended, or a timing simulation of the post implementation design to verify circuit operation after the worst-case placed and routed delays are calculated. The default mode is **behavioral**.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example launches ISim in timing mode:

```
launch_isim -mode timing
```

See Also

create_fileset

launch_modelsim

Launch simulation using ModelSim simulator.

Syntax

```
launch_modelsim [-simset arg] [-mode arg] [-type arg]
[-noclean_dir] [-scripts_only] [-install_path arg] [-quiet]
[-verbose]
```

Returns

Nothing

Usage

Name	Description
[-**simset**]	Name of the simulation fileset
[-**mode**]	Simulation mode. Values: behavioral, post-synthesis, post-implementation Default: behavioral
[-**type**]	Netlist type. Values: functional, timing. This is only applicable when mode is set to post-synthesis or post-implementation
[-**noclean_dir**]	Do not remove simulation run directory files
[-**scripts_only**]	Only generate scripts
[-**install_path**]	Custom ModelSim installation directory path
[-**quiet**]	Ignore command errors
[-**verbose**]	Suspend message limits during command execution

Categories

ToolLaunch, Simulation

Description

Launch the Mentor Graphics ModelSim or Questa Advanced Simulator tool. The specified simulator must be installed, and appear in your $PATH in order to be properly invoked when launching simulation.

To launch ModelSim or Questa, you must first set the target simulator property for the project:

```
set_property target_simulator ModelSim [current_project]
```

In order to support the use of ModelSim/Questa you must compile the Xilinx simulation libraries for use with the target simulator using the **compile_simlib** command. After the libraries are compiled, the simulator will reference these compiled libraries using the `modelsim.ini` file. The `modelsim.ini` file is the default initialization file and contains control variables that specify reference library paths, optimization, compiler and simulator settings. The `modelsim.ini` is located as follows:

- The path specified by **-directory** argument at the time **compile_simlib** is run.

- The path defined by `MODELSIM` environment variable.

- The path defined by `MGC_WD` environment variable.

- The simulation run directory of the project.

Note If the `modelsim.ini` file is not found at any of these locations a warning message is returned by the simulator.

The command returns the transcript of the simulator.

Arguments

-simset - (Optional) Name of the simulation fileset containing the simulation test benches and sources to be used during simulation. If not specified, the current simulation fileset is used.

-noclean_dir - (Optional) Do not remove simulation run directory files prior to launching the simulator. However, some of the files generated for use by the simulator will be overwritten or updated by re-launching the simulator. The default is to remove the simulation run directory before launching the simulator.

-mode [behavioral | timing] - (Optional) Simulation mode. Specifies either a behavioral simulation of the design to verify syntax and confirm that the design performs as intended, or a timing simulation of the post implementation design to verify circuit operation after the worst-case placed and routed delays are calculated. The default mode is **behavioral**.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example launches the ModelSim simulator in timing mode:

```
launch_modelsim -mode timing
```

See Also

set_property

launch_runs

Launch a set of runs.

Syntax

```
launch_runs [-jobs arg] [-scripts_only] [-all_placement]
[-dir arg] [-to_step arg] [-next_step] [-host args]
[-remote_cmd arg] [-email_to args] [-email_all]
[-pre_launch_script arg] [-post_launch_script arg]
[-force] [-quiet] [-verbose] runs...
```

Returns

Nothing

Usage

Name	Description
[-**jobs**]	Number of jobs Default: 1
[-**scripts_only**]	Only generate scripts
[-**all_placement**]	Export all fixed and non-fixed placement to ISE (by default only fixed placement will be exported)
[-**dir**]	Launch directory
[-**to_step**]	Last Step to run. Ignored when launching multiple runs. Not valid with -next_step
[-**next_step**]	Run next step. Ignored when launching multiple runs. Not valid with -to_step.
[-**host**]	Launch on specified remote host with a specified number of jobs. Example: -host {machine1 2} -host {machine2 4}
[-**remote_cmd**]	Command to log in to remote hosts Default: ssh -q -o BatchMode=yes
[-**email_to**]	List of email addresses to notify when jobs complete
[-**email_all**]	Send email after each job completes
[-**pre_launch_script**]	Script to run before launching each job
[-**post_launch_script**]	Script to run after each job completes
[-**force**]	Run the command, even if there are pending constraint changes, which will be lost (in a Partial Reconfig design)
[-**quiet**]	Ignore command errors
[-**verbose**]	Suspend message limits during command execution
runs	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* - The number of parallel jobs to run on the local host. 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 for implementation. As a default, the tool will export only the fixed or user-assigned placement for implementation.

-dir *arg* - The directory for the tool to write run results into. A separate folder for each run is created under the specified directory. As a default the tool will write the results of each run 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 place_design step, and then stop. This will allow you to look at specific stages of a run without completing the entire run.

Valid values for ISE implementation are: NGDBuild, MAP, PAR, TRCE, XDL.

-next_step - Continue a prior run from the step at which it was stopped. 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, and are ignored when launching multiple runs

-host *args* - 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.

Note This argument is supported on the Linux platform only.

-remote_cmd *arg* - 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* - A script to run before launching each job.

-post_launch_script *arg* - A script to run after completion of all jobs.

-force - Launch the run regardless of any pending constraint changes for Partial Reconfiguration designs.

Note This argument applies only to Partial Reconfiguration projects. Any pending constraint changes will be lost to the specified runs.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

runs - The names of synthesis and 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_sdk

Launch Xilinx Software Development Kit (SDK).

Syntax

launch_sdk [**-bit** *arg*] [**-bmm** *arg*] [**-workspace** *arg*] [**-hwspec** *arg*] [**-quiet**] [**-verbose**]

Returns

Nothing

Usage

Name	Description
[**-bit**]	Specify the bitstream file for FPGA programming
[**-bmm**]	Specify the BMM file for BRAM initialization
[**-workspace**]	Specify the workspace directory for SDK projects
[**-hwspec**]	Specify the hardware platform specification file (.xml)
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

[ToolLaunch](), [XPS]()

Description

Launch the Software Development Kit (SDK) to design the software for embedded processor sources in your project.

This command follows the **export_hardware** command, which exports the embedded processor hardware specification file (system.xml) for use by SDK. By default, the export_hardware command will write the hardware specification file (.xml) for the specified embedded processors to the *project_name*.sdk/SDK/SDK_Export/hw directory, to a file named after the embedded processor in the design, with the .XML extension.

The command returns a transcript of the SDK tool launch.

Arguments

-bit *arg* - (Optional) Specify the bitstream file for FPGA programming.

-bmm *arg* - (Optional) Specify the BMM file for BRAM initialization.

-workspace *arg* - (Optional) Specify the workspace directory for SDK projects. This is the folder in which your software projects are stored.

-hwspec *arg* - (Optional) The hardware platform specification file (.xml) for the Embedded Processor design. This is the file exported by the export_hardware command, or is the system.xml file found in the sources/edk directory of the project.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example launches SDK, loading the specified hardware specification file for the project, and indicates the workspace to use:

```
launch_sdk -hwspec C:/Data/export_sdk/hw/robot.xml -workspace C:/Data/sdk_work/
```

See Also

export_hardware

launch_xpa

Launch XPower Analyzer tool.

Syntax

launch_xpa [**-run** *arg*] [**-more_options** *arg*] [**-mapped_ncd**] [**-quiet**] [**-verbose**]

Returns

Nothing

Usage

Name	Description
[**-run**]	Implemented run to launch XPA with
[**-more_options**]	More command line options
[**-mapped_ncd**]	Use Mapped NCD
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

ToolLaunch

link_design

Open a netlist design.

Syntax

```
link_design [-name arg] [-part arg] [-constrset arg] [-top arg]
[-mode arg] [-quiet] [-verbose]
```

Returns

Design object

Usage

Name	Description
[**-name**]	Design name
[**-part**]	Target part
[**-constrset**]	Constraint fileset to use
[**-top**]	Specify the top module name when the structural netlist is Verilog
[**-mode**]	The design mode. Values: default, out_of_context Default: default
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Tools

Description

Opens a new or existing Netlist design, linking the netlists and constraints with the target part to create the design. This can also be accomplished with the **open_run** command.

The design_mode property for the current source fileset must be defined as GateLvl in order to open a Netlist design. If not, you will get the following error:

```
ERROR: The design mode of 'sources_1' must be GateLvl.
```

Arguments

-name *arg* - The name of a new or existing Netlist design.

-part *arg* - The Xilinx device to use when creating a new design. If the part is not specified the default part will be used.

-constrset *arg* - 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* - The top module of the design hierarchy of the netlist.

-mode [default | out_of_context] - (Optional) If you have synthesized a block, and disabled IO buffer insertion, you can load the resulting EDIF into the Vivado Design Suite using **-mode out_of_context**. This enables implementation of the module without IO buffers, prevents optimization due to unconnected inputs or outputs, and adjusts DRC rules appropriately for the design. Refer to the *Vivado Design Suite User Guide: Hierarchical Design (UG905)* for more information.

Note -mode is not intended for use in the PlanAhead tool.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following creates a new Netlist design called Net1:

```
link_design -name Net1
```

Note The default source set, constraint set, and part will be used in this example.

The following example opens a Netlist design called Net1, and specifies the constraint set to be used:

```
link_design -name Net1 -constrset con1
```

See Also

open_run

list_param

Get all parameter names.

Syntax

list_param [**-quiet**] [**-verbose**]

Returns

List

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

PropertyAndParameter

Description

Gets a list of user-definable configuration parameters. These parameters configure a variety of settings and behaviors of the tool. For more information on a specific parameter use the **report_param** command, which returns a description of the parameter as well as its current value.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example returns a list of all user-definable parameters:

```
list_param
```

See Also

- get_param
- report_param
- reset_param
- set_param

list_property

List properties of object.

Syntax

list_property [**-class** *arg*] [**-quiet**] [**-verbose**] [*object*]

Returns

List of property names

Usage

Name	Description
[**-class**]	Object type to query for properties. Ignored if object is specified.
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*object*]	Object to query for properties

Categories

Object, PropertyAndParameter

Description

Gets a list of all properties on a specified 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* - The class of object for which to list the properties.

Note When both **-class** and *object* are specified, the properties of the specific object are returned.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

object - The single object on which to report properties.

Note If you specify multiple objects you will get an error.

Examples

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 [-default] [-class arg] [-quiet]
[-verbose] name [object]
```

Returns

List of property values

Usage

Name	Description
[-default]	Show only the default value.
[-class]	Object type to query for legal property values. Ignored if object is specified.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
name	Name of property whose legal values is to be retrieved
[object]	Object to query for legal properties values

Categories

Object, PropertyAndParameter

Description

Gets a list of valid values for an enumerated type property of either a class of objects or a specific object.

Note The command cannot be used to return valid values for properties other than enum properties. The **report_property** command will return the type of property to help you identify enum properties.

Arguments

-defaut - Return the default value for the specified class of objects.

-class *arg* - The class of object to query. The class of object can be used in place of an actual object.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

name - 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 valid values of the specified property will be returned.

object - An object to query. An actual object can be used in place of the **-class** argument to specify the type of object to query.

Examples

The following example returns the list of valid 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]
```

The following example returns the default value for the specified property by using the current design as a representative of the design class:

```
list_property_value -default BITSTREAM.GENERAL.COMPRESS [current_design]
```

See Also

- create_property
- current_design
- get_cells
- get_property
- list_property
- report_property
- reset_property
- set_property

list_targets

List applicable targets for the specified source.

Syntax

```
list_targets [-quiet] [-verbose] files
```

Returns

List of targets

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
files	Source file for which the targets needs to be listed

Categories

Project, XPS

Description

List the targets that are available for a specified IP core, DSP module, Embedded Processor source, or Block Diagram.

Use the **generate_targets** command to generate the listed targets.

The command returns the list of available targets. If no targets are available for the specified file objects, nothing is returned.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

files - A files object that contains the list of source files to evaluate.

Note Use **get_files** to specify a files object, rather than specifying a file name.

Examples

The following example lists the available targets for any DSP modules in the design:

```
list_targets [get_files *.mdl]
```

See Also

- create_sysgen
- create_xps
- generate_target
- get_files
- import_ip
- read_ip

load_reconfig_modules

Load specific Reconfigurable Modules or all modules from a given run.

Syntax

```
load_reconfig_modules [-force] [-run arg] [-quiet] [-verbose]
[reconfig_modules ]
```

Returns

Nothing

Usage

Name	Description
[**-force**]	Run the command, even if there are pending constraint changes, which will be lost
[**-run**]	Run to load Reconfigurable Modules from
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*reconfig_modules*]	Reconfigurable Module(s) to load

Categories

PartialReconfiguration

Description

Load the netlist and constraint files of the specified reconfigurable module (RM), or all modules from a given run, into the current design.

You can create the RM using the **create_reconfig_module** command, and then load it into the design later.

Although a single partition cell can contain multiple RMs, only one can be loaded into the current design as the active module for that cell. Use the **load_reconfig_modules** command to change the active module for a specific partition cell.

The combination of active modules in the design, and all other logic, is called a configuration of the design.

Arguments

-force - (Optional) Force the load of the RM even when there are outstanding design changes to be saved. If there are pending design changes, and **-force** is not specified, an error will be returned.

Note You should use **save_design** prior to using **load_reconfig_modules**.

-run *arg* - (Optional) The run from which to load RMs. The specific implementation of the modules for the given run are loaded.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

reconfig_modules - (optional) The names of reconfigurable modules to load.

Note You must use either the **-run** argument, or specify one or more RMs to load

Examples

The example code below loads the netlist and constraints of the reconfigurable module named BramFirst for the specified partition cell, then loads the BramSecond to make it the active module for the same cell:

```
load_reconfig_modules -reconfig_modules U2_RP_Count:BramFirst
load_reconfig_modules -reconfig_modules U2_RP_Count:BramSecond
```

Note The reconfigurable module is specified by the hierarchical name *<cell>:<name>*.

See Also

- create_reconfig_module
- delete_reconfig_module
- get_reconfig_modules

make_diff_pair_ports

Make differential pair for 2 ports.

Syntax

make_diff_pair_ports [**-quiet**] [**-verbose**] *ports...*

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
ports	Ports to join

Categories

PinPlanning

Description

Joins two existing ports to create a differential pair.

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

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

ports - Two port objects to join as a differential pair. The first port specified should be the positive side of the diff pair.

Examples

The following example joins the two specified ports to create a differential pair:

```
make_diff_pair_ports PORT_P PORT_N
```

See Also

- create_interface
- create_port

make_wrapper

Generate HDL wrapper for the specified source.

Syntax

```
make_wrapper [-top] [-testbench] [-inst_template] [-fileset arg]
[-import] [-quiet] [-verbose] files
```

Returns

Nothing

Usage

Name	Description
[**-top**]	Create a top-level wrapper for the specified source
[**-testbench**]	Create a testbench for the specified source
[**-inst_template**]	Create an instantiation template for the specified source. The template will not be added to the project and will be generated for reference purposes only.
[**-fileset**]	Fileset name
[**-import**]	Import generated wrapper to the project
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
files	Source file for which the wrapper needs to be generated

Categories

Project, XPS, SysGen

Description

Create a Verilog or VHDL wrapper for instantiating an IP core, DSP module, or Embedded Processor sub-design into a project.

Note The wrapper is generated in Verilog or VHDL according to the TARGET_LANGUAGE property on the project

The command returns information related to the creation of the wrappers.

Arguments

-top - (Optional) Create a top-level Verilog or VHDL wrapper for the specified source. The wrapper instantiates the DSP module or Embedded Processor sub-design as the top-level of the design hiearchy.

-testbench - (Optional) Create a simulation testbench for the specified DSP module or Embedded Processor sub-design. This includes the DUT module instantiation.

-inst_template - (Optional) Create an instantiation template for the specified source. The template will not be added to the project and will be generated for reference purposes only. The instantiation template can be cut and paste into another RTL file to create an instance of the module in the hierarchy.

-fileset - (Optional) Specify the fileset to add the wrapper to when importing into the project.

-import - (Optional) Import the wrapper file into the project, adding it to the appropriate fileset.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

files - Specify the files to generate wrappers for.

Examples

The following example create the instantiation template to integrate the specified Embedded Processor source into the design hierarchy:

```
make_wrapper -inst_template -fileset [get_filesets sources_1] \
    [get_files C:/Data/edk/xpsTest1/xpsTest1.xmp]
```

See Also

- add_files
- create_sysgen
- create_xps
- generate_target
- import_ip
- list_targets
- read_ip

mark_objects

Mark objects in GUI.

Syntax

```
mark_objects [-rgb args] [-color arg] [-quiet]
[-verbose] objects
```

Returns

Nothing

Usage

Name	Description
[-rgb]	RGB color index list
[-color]	Valid values are red green blue magenta yellow cyan and orange
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
objects	Objects to mark

Categories

GUIControl

Description

Marks specified objects in 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 Use only one color option. If both color options are specified, **-rgb** takes precedence over **-color**

Arguments

-rgb *args* - 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* - The color to use for marking the specified object or objects. Supported colors are: red, green, blue, magenta, yellow, cyan, and orange.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

objects - One or more objects to be marked.

Examples

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_example_project

Open the example project for the indicated IP.

Syntax

```
open_example_project [-dir arg] [-force] [-in_process] [-quiet]
[-verbose] objects...
```

Returns

The Project that was opened

Usage

Name	Description
[-dir]	Path to directory where example project will be created
[-force]	Overwrite an example project if it exists
[-in_process]	Open the example project in the same process
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
objects	The objects whose example projects will be opened

Categories

Project, IPFlow

Description

Open an example project for the specified IP cores. The example project can be used to explore the features of the IP core in a stand-alone project, instead of integrated into the current project.

Arguments

-dir *arg* - (Optional) Specifies the path to the directory where the example project will be written.

-force - (Optional) Force the opening of a new example project, overwriting an existing example project at the specified path.

-in_process - (Optional) Open the example project in the same tool process as the current project. As a default, without this argument, a new process instance of the tool will be launched for the example project.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

objects - The IP cores to open example projects for.

Examples

The following generates the target data for the example project, then opens the example project for the specified IP core:

```
generate_target {example} [get_ips blk_mem*]
open_example_project -force [get_ips blk_mem*]
```

Note The Example target data must be generated prior to using the **open_example_project** command. This will create a Tcl script to open and configure the specified IP core

See Also

- create_ip
- generate_target
- get_ips
- import_ip

open_io_design

Open an IO design.

Syntax

open_io_design [**-name** *arg*] [**-part** *arg*] [**-constrset** *arg*] [**-quiet**] [**-verbose**]

Returns

Design object

Usage

Name	Description
[**-name**]	Design name
[**-part**]	Target part
[**-constrset**]	Constraint fileset to use
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

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. If not, you will get the following error:

```
ERROR: The design mode of 'sources_1' must be PinPlanning
```

Arguments

-name *arg* - The name of a new or existing I/O Pin Planning design.

-part *arg* - The Xilinx device to use when creating a new design. If the part is not specified the default part will be used.

-constrset *arg* - 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

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_project

Open a PlanAhead project file (.ppr).

Syntax

```
open_project [-read_only] [-quiet] [-verbose] file
```

Returns

Opened project object

Usage

Name	Description
[**-read_only**]	Open the project in read-only mode
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
file	Project file to be read

Categories

Project

Description

Opens a project file (.ppr) 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 - (Optional) Open the project in read only mode. You will not be able to save any modifications to the project unless you use the **save_project_as** command to save the project to a new editable project.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

file - The project file to open. You must include both the path to the file and the `.ppr` file extension.

Examples

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 tool to recognize it as a project file. The path to the file must be specified along with the project file name or the 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] [-verbose]
```

Returns

Design object

Usage

Name	Description
[-name]	Design name
[-part]	Target part
[-constrset]	Constraint fileset to use
[-top]	Specify the top module name
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Project

Description

Opens a new or existing RTL source design.

Note 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* - The name of a new or existing RTL design.

-part *arg* - The Xilinx device to use when creating a new design. If the part is not specified the default part will be used.

-constrset *arg* - 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

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

launch_runs

open_run

Open a run into a netlist or implementation design.

Syntax

open_run [**-name** *arg*] [**-quiet**] [**-verbose**] *run*

Returns

Design object

Usage

Name	Description
[**-name**]	Design name
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
run	Run to open into the design

Categories

Project

Description

Opens the specified synthesis run into a Netlist Design or implementation run into an Implemented Design. The run properties defining the target part and constraint set are combined with the synthesis or implementation results to create the design view in the tool.

Arguments

-name - (Optional) The name of the design to open.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

run - Specifies the run name of the synthesis or implementation run to open. The run must have completed synthesis or implementation before it can be opened as a design.

Note If you attempt to open a run that has not been launched the tool will return an error.

Examples

The following command opens the specified synthesis run into a Netlist Design named synthPass1:

```
open_run -name synthPass1 synth_1
```

The following opens an Implemented Design for impl_1:

```
open_run impl_1
```

See Also

launch_runs

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] [-verbose] cell_site_list ...
```

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
cell_site_list	a list of cells and sites in the interleaved order

Categories

Floorplan

Description

Places cells onto device resources of the target part. Cells can be placed onto specific BEL sites (e.g. **SLICE_X49Y60/A6LUT**), or into available SLICE resources (e.g. **SLICE_X49Y60**). If you specify the SLICE but not the BEL the tool will determine an appropriate BEL within the specified SLICE if one is available.

When placing a cell onto a specified site, the site must not be currently occupied, or an error will be returned: "Cannot set site and bel property of instances. Site SLICE_X49Y61 is already occupied."

You can test if a site is occupied by querying the IS_OCCUPIED property of a BEL site:

```
get_property IS_OCCUPIED [get_bels SLICE_X48Y60/D6LUT]
```

Note The IS_OCCUPIED property of a SLICE only tells you if some of the BELs within the SLICE are occupied; not whether or not the SLICE is fully occupied.

This command can be used to place cells, or to move placed cells from one site on the device to another site. The command syntax is the same for placing an unplaced cell, or moving a placed cell.

When moving a placed cell, if you specify only the SLICE for the site, the tool will attempt to place the cell onto the same BEL site in the new SLICE as it currently is placed. For instance moving a cell from the B6LUT, by specifying a new SLICE, will cause the tool to attempt to place the cell onto the B6LUT in the new SLICE. If this BEL site is currently occupied, an error is returned.

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

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

cell_site_list - Specifies a list of cells and sites as {*cell_name site*}. The cell name is listed first, followed the BEL site or SLICE to place the cell onto. If the site is specified as a SLICE, the tool will select an available BEL within the SLICE. Multiple cells can be placed onto multiple sites by repeating the cell/site pair multiple times as needed:

{*cell_name1 site1 cell_name2 site2 cell_name3 site3 ... cell_nameN siteN*}.

Examples

The following example places the specified cell onto the specified BEL site:

```
place_cell div_cntr_reg_inferredi_4810_15889 SLICE_X49Y60/D6LUT
```

The following example places the specified cell into the specified SLICE:

```
place_cell div_cntr_reg_inferredi_4810_15889 SLICE_X49Y61
```

Note The tool will select an appropriate BEL site if one is available. If no BEL is available, and error will be returned

The following example places multiple cells onto multiple sites:

```
place_cell  { \
cpuEngine/cpu_iwb_adr_o/buffer_fifo/i_4810_17734 SLICE_X49Y60/A6LUT \
cpuEngine/or1200_cpu/or1200_mult_mac/i_4775_15857 SLICE_X49Y60/B6LUT \
cpuEngine/cpu_iwb_adr_o/buffer_fifo/xlnx_opt_LUT_i_4810_18807_2 SLICE_X49Y60/C6LUT }
```

See Also

unplace_cell

place_pblocks

Run the pblocks Placer.

Syntax

```
place_pblocks [-effort arg] [-utilization arg] [-quiet]
[-verbose] pblocks...
```

Returns

Nothing

Usage

Name	Description
[-effort]	Placer effort level (per pblock) Values: LOW, MEDIUM, HIGH Default: HIGH
[-utilization]	Placer utilization (per pblock)
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
pblocks	List of pblocks to place

Categories

Floorplan

Description

Places 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* - Effort level that the Pblock placer should use in placing each Pblock onto the fabric. Valid values are LOW, MEDIUM, HIGH, with the default being HIGH.

-utilization *arg* - 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

pblocks - One or more Pblocks to be placed onto the fabric of the FPGA.

Examples

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]
[-verbose] [ports...]
```

Returns

Nothing

Usage

Name	Description
[**-skip_unconnected_ports**]	Do not place unconnected ports
[**-check_only**]	Only check IO/Clock placement DRCs
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*ports*]	Ports to place (if omitted, all ports will be placed)

Categories

PinPlanning

Description

Automatically places ports on an available I/O or clocking site.

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

Arguments

-skip_unconnected_ports - (Optional) Do not place unconnected ports

-check_only - (Optional) Check only I/O or clocking design rule checks (DRCs) during placement.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

ports - (Optional) The names of the ports to be placed. If no *ports* are specified, all ports will be placed.

Examples

The following example automatically places all ports in the design, and only checks I/O and clock DRCs:

```
place_ports -check_only
```

See Also

- create_interface
- create_port
- make_diff_pair_ports
- remove_port

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] [-verbose] run
```

Returns

Nothing

Usage

Name	Description
[-partition_names]	List of Partitions to be promoted
[-promote_dir]	Promote path
[-description]	Promote description
[-no_state_update]	Do not automatically update the state of promoted Partitions to Import
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
run	Implemented run to be promoted

Categories

PartialReconfiguration, Partition

Description

Promotes previously implemented partitions to make them available for import and reuse in other designs.

Arguments

-partition_names *arg* - (Optional) The partition or partitions to be promoted.

-promote_dir *arg* - (Optional) The path and directory into which to write the partition data.

Note If the specified directory already exists, the **promote_run** command will overwrite it without warning

Note If the path is not specified as part of the directory name, the tool will create a directory in your home directory.

• For Windows: %APPDATA%/Xilinx/PlanAhead

• For Linux: $HOME/.Xilinx/PlanAhead

-description *{text description}* - (Optional) A description of the partition being promoted. Enclose *text* in braces {}. This description can be displayed or queried when using the partition in another design.

-no_state_update - (Optional) Do not automatically update partition states. By default the tool will notify you of any updates to a promoted partition. After promotion, the partition state will change to import for promoted partitions.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

run - The implemented run to be promoted.

Examples

The following example promotes the synthesis run for the usbEngine0 and usbEngine1 partitions, and promotes the implementation run for usbEngine0:

```
promote_run -run synth_1 -partition_names {usbEngine0 usbEngine1} -promote_dir \
C:/Data/partition/DP_RTL.promote/Xsynth_1
promote_run -run impl_1 -partition_names usbEngine0 -promote_dir \
C:/Data/partition/DP_RTL.promote/Ximpl_1 -description {Implementation of USB0}
```

See Also

- config_partition
- load_reconfig_modules

read_chipscope_cdc

Import ChipScope Core Inserter CDC file.

Syntax

```
read_chipscope_cdc [-quiet] [-verbose] file
```

Returns

Nothing

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
file	ChipScope CDC file name

Categories

FileIO, ChipScope

Description

Reads 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 be written by the tool, or from an ISE 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

file - The path and filename of the CDC file to read into. The CDC file defines the debug core to insert, the signals to probe, and the clock domains for those signals.

Examples

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_diff_pairs**] [**-quiet**] [**-verbose**] *file*

Returns

Nothing

Usage

Name	Description
[**-quiet_diff_pairs**]	Suppress warnings about differential pair inference when importing I/O ports
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
file	Pin Planning CSV file

Categories

FileIO

Description

Imports port definition and package pin placement information from a comma separated value (CSV) file.

The port definitions in a CSV file can be imported into an I/O Pin Planning project. In a Pin Planning project, importing a CSV file replaces the current port definitions. Any ports in the design that are not found in the imported CSV file will be removed.

In all other projects the port definitions are defined in the source design data, however package pin assignments and port attributes can be read from the specified CSV file.

The ports read from the CSV file can not have spaces in the name, or the tool will return an error. The specific format and requirements of the CSV file are described in the *PlanAhead Users Guide* (UG632.pdf).

Arguments

-quiet_diff_pairs - (Optional) The tool transcripts messages related to pins that may be inferred as differential pairs when importing the CSV file. This option suppresses messages related to inferring differential pairs.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

file - The file name of the CSV file to be imported.

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

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, and infers any differential pairs in the CSV file:

```
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
infer_diff_pairs -filetype 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
- infer_diff_pairs
- open_io_design
- set_property
- write_csv

read_edif

Read one or more EDIF or NGC files.

Syntax

```
read_edif [-quiet] [-verbose] files
```

Returns

List of file objects that were added

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
files	EDIF or NGC file name(s)

Categories

FileIO

Description

Imports an EDIF or NGC netlist file into the Design Source fileset of the current project.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

files - The name of the EDIF or NGC files to be imported.

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

Examples

The following example imports an EDIF file into the open project:

```
read_edif C/Data/bft_top.edf
```

See Also

write_edif

read_ip

Read one or more IP files.

Syntax

read_ip [**-quiet**] [**-verbose**] *files*

Returns

List of IP file objects that were added

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
files	IP file name(s)

Categories

FileIO, IPFlow

Description

Read the specified list of IP files (XCO) and add them to the design and the current fileset.

An XCO file is a CORE Generator 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 XCO files are used to recreate the core in the current project.

Note The PlanAhead tool reads IP from the ChipScope tool. It does not support reading Vivado native IP (XCI) files

Files are added by reference into the current project, just as in the **add_files** command. Use the **import_ip** command to add the IP cores and import the files into the local project directory.

The command returns the list of files read.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

files - The list of IP files (XCO) to read into the current project.

Examples

The following example reads the specified IP files:

```
read_ip C:/Data/char_fifo.xco
```

See Also

- add_files
- import_ip

read_pxml

Import Partition definitions from a PXML file.

Syntax

```
read_pxml [-quiet] [-verbose] file
```

Returns

Nothing

Usage

Name	Description
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
file	PXML file name

Categories

FileIO

Description

Imports 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 tool during synthesis or implementation. You can also create one by hand or by using an XML template provided in the 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

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

file - The name of the PXML file. The file must be named xpartition.pxml, or you will get an error when trying to read the file.

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

Examples

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**] [**-verbose**] *name file*

Returns

Nothing

Usage

Name	Description
[**-cell**]	Interpret names in the report file as relative to the specified cell
[**-pblock**]	Interpret names in the report file as relative to the specified pblock
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
name	Name for the set of results
file	Name of the Trace import file

Categories

FileIO

Description

Imports 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* - (Optional) 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* - (Optional) The name of a Pblock in the current design. The timing paths will be imported relative to the specified block.

name - 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 - The file name of the TWX file to be imported.

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

Examples

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 [**-cells** *args*] [**-ref** *arg*] [**-quiet_diff_pairs**] [**-quiet**]
[**-verbose**] *file*

Returns

List of added files

Usage

Name	Description
[**-cells**]	Import constraints for these cells
[**-ref**]	Import constraints for this ref
[**-quiet_diff_pairs**]	Suppress warnings about differential pair inference when importing I/O ports
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
file	Input UCF file name

Categories

FileIO

Description

Imports 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.

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.

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.

Arguments

file - The file name of the UCF file to be imported.

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

-cell *arg* - 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

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
- infer_diff_pairs
- write_ucf

read_verilog

Read one or more Verilog files.

Syntax

read_verilog [**-library** *arg*] [**-sv**] [**-quiet**] [**-verbose**] *files...*

Returns

List of file objects that were added

Usage

Name	Description
[**-library**]	Library name Default: work
[**-sv**]	Enable system verilog compilation
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
files	Verilog file name(s)

Categories

FileIO

Description

Reads Verilog source files. This command is 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.

Arguments

-library *arg* - The library the Verilog file should reference. The default Verilog library is work.

files - 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 tool will search for the specified file in the current working directory and then in the directory from which the tool was launched.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

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
```

See Also

- add_files
- read_vhdl

read_vhdl

Read one or more VHDL files.

Syntax

read_vhdl [**-library** *arg*] [**-quiet**] [**-verbose**] *files*

Returns

List of file objects that were added

Usage

Name	Description
[**-library**]	VHDL library Default: work
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
files	VHDL file name(s)

Categories

FileIO

Description

Reads a VHDL source file. This command is 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* - (Optional) The library the VHDL file should reference. The default VHDL library is work.

file - Filename of the VHDL file to be read.

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

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

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_verilog

read_xdl

Import placement information from a file.

Syntax

read_xdl [**-pblock** *arg*] [**-cell** *arg*] [**-quiet**] [**-verbose**] *file*

Returns

Nothing

Usage

Name	Description
[**-pblock**]	Import placement for this pblock
[**-cell**]	Import placement for this cell
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
file	Placement file name

Categories

FileIO

Description

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* - (Optional) 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* - (Optional) 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

file - The file name of the XDL file to be imported.

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

Examples

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**] [**-verbose**]

Returns

With -list, the list of redoable tasks

Usage

Name	Description
[**-list**]	Show a list of redoable tasks
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

GUIControl

Description

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 - Get the list of commands that can be redone. When you use the **undo** command, the tool will step backward through a list of commands. The **redo** command can then be used to redo those commands.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example returns a list of commands that can be redone:

```
redo -list
```

See Also

- undo
- startgroup
- endgroup

refresh_design

Refresh the current design.

Syntax

refresh_design [**-part** *arg*] [**-quiet**] [**-verbose**]

Returns

Nothing

Usage

Name	Description
[**-part**]	Target part
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

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* - (Optional) The new target part for the design when it is reloaded. This overrides the constraint file part specified in the project data on the hard drive.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

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
```

Note 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] [-verbose] [files...]
```

Returns

List of file objects that were imported

Usage

Name	Description
[**-force**]	Force a reimport to happen even when the local files may be newer
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*files*]	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

Reimports project files. This updates the local project files from the original referenced source files.

Arguments

-force - (Optional) Reimport files even when the local project files may be newer than their referenced source files.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

files - (Optional) List of files to reimport. If no files are specified, all files in the project that are out-of-date, will be reimported. If you use **-force** and specify no files, all files in the project will be reimported.

Examples

The following example reimports 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, but only if the original source file is newer than the local project file:

```
reimport_files C:/Data/FPGA_Design/source1.v C:/Data/FPGA_Design/source2.vhdl
```

See Also

- add_files
- import_files

remove_cells_from_pblock

Remove cells from a Pblock.

Syntax

remove_cells_from_pblock [**-quiet**] [**-verbose**] *pblock cells...*

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
pblock	Pblock to remove cells from
cells	Cells to remove

Categories

Floorplan, XDC

Description

Removes the specified logic instances from a Pblock. Cells are added to a Pblock with the **add_cells_to_pblock** command.

Note 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

pblock - The name of the Pblock from which to remove the specified instances.

cells - One or more cell objects to remove from the specified Pblock.

Examples

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_files

Remove files or directories from a fileset.

Syntax

remove_files [**-fileset** *arg*] [**-quiet**] [**-verbose**] [*files*...]

Returns

List of files that were removed

Usage

Name	Description
[**-fileset**]	Fileset name
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*files*]	Name of the files and/or directories to remove

Categories

Project, Simulation

Description

Removes the specified files from the current or specified fileset.

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

Arguments

-fileset *arg* - The name of the fileset from which to remove the specified files. As a default, the files will be removed from the current fileset as defined by the current_fileset command.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

files - The name of the files, or directories, to remove from the project.

Examples

The following example removes the file named `C:/Design/top.xdc` from the constraint set `constrs_1`:

```
remove_files -fileset constrs_1 C:/Design/top.xdc
```

Multiple files can be specified as follows:

```
remove_files -fileset sim_1 top_tb1.vhdl top_tb2.vhdl
```

See Also

- add_files
- current_fileset

remove_pin

Remove pins from the current design.

Syntax

remove_pin [**-quiet**] [**-verbose**] *pins...*

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
pins	List of pins to remove

Categories

Netlist

Description

Remove pins from the current netlist in either an open Synthesized or Implemented design.

Netlist editing changes the in-memory view of the netlist in the current design. It does not change the files in the source fileset, or change the persistent design on the disk. Changes made to the netlist may be saved to a design checkpoint using the **write_checkpoint** command, or may be exported to a netlist file such as Verilog, VHDL, or EDIF, using the appropriate **write_*** command.

Note Netlist editing is not allowed on the elaborated RTL design.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

pins - List of pins to remove from the netlist. The pins must be specified hierarchically by the cell instance the pin is found on.

Examples

The following example removes the fftEngine from the in-memory netlist of the current design:

```
remove_cell fftEngine
```

See Also

- write_edif
- write_verilog
- write_vhdl

remove_port

Remove the given list of top ports from the netlist.

Syntax

remove_port [**-quiet**] [**-verbose**] *ports...*

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
ports	Ports and/or bus ports to remove

Categories

PinPlanning

Description

Removes the specified ports or buses.

The **remove_port** command will remove ports that have been added with the **create_port** command, but cannot delete ports that are defined in the RTL or netlist design.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

ports - One or more names of ports to remove.

Examples

The following example deletes the specified port:

```
remove_port PORT0
```

The following example deletes the two specified ports of a bus:

```
remove_port BUS[1] BUS[2]
```

The following example deletes both the N and P sides of a differential pair port:

```
remove_port D_BUS_P[0]
```

Note Deleting either the N or the P side of a differential pair will also delete the other side of the pair.

See Also

- create_port
- create_interface
- place_ports

rename_ref

Rename a cell ref.

Syntax

```
rename_ref [-ref arg] [-to arg] [-prefix_all arg] [-quiet]
[-verbose]
```

Returns

Nothing

Usage

Name	Description
[-ref]	Cell ref to rename
[-to]	New name
[-prefix_all]	Rename all eligible hierarchical cell refs in the current design. Construct the new name using the given prefix plus the original name
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Netlist

Description

Rename the reference name of a single non-primitive cell, or apply a reference prefix to all non-primitive cells in the current synthesized or implemented design.

This command provides a mechanism to change the non-primitive reference names in the current design so that they do not collide with the reference names in another design. This lets two modules or designs be synthesized or simulated together, while avoiding any name collisions between the two designs.

This command returns nothing when renaming the reference a single cell, and returns the number of cells renamed when used with **-prefix_all**. If the command fails, an error is returned.

Arguments

-ref *arg* - (Optional) Specify the current reference name of a non-primitive cell.

-to *arg* - (Optional) Change the reference name to the specified value.

-prefix_all *arg* - (Optional) Apply the specified prefix to the reference names of all non-primitive cells in the current design, except the top module.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example changes the specified reference name to the value indicated:

```
rename_ref -ref usbf_top -to MOD1_usbf_top
```

The following example applies the specified reference name prefix to all non-primitive cells in the current design:

```
rename_ref -prefix_all MOD1_
```

See Also

- launch_isim
- launch_runs
- write_edif
- write_verilog
- write_vhdl

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] [-verbose] files...
```

Returns

Nothing

Usage

Name	Description
[**-fileset**]	Fileset to reorder
[**-before**]	Move the listed files before this file
[**-after**]	Move the listed files after this file
[**-front**]	Move the listed files to the front (default)
[**-back**]	Move the listed files to the back
[**-auto**]	Automatically re-orders the given fileset
[**-disable_unused**]	Disables all files not associated with the TOP design unit
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
files	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 reorders the files based on the requirements of the current top module in the design.

Arguments

-fileset *arg* - The fileset in which to reorder files. 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.

-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.

-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 - Enable automatic reordering based on the hierarchy requirements of the current top-module in the project. Often used after changing the top module with the "**set_property top**" command.

-disable_unused - Disable any files not currently used by the hierarchy based on the top-module. Often used after changing the top module with the "**set_property top**" command.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

files - One or more files to relocate in the fileset. Files must be specified by their full path name in the fileset, and are reordered in the order they are specified.

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 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 automatically reorders and disables 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_carry_chains

Report carry chains.

Syntax

```
report_carry_chains [-file arg] [-append] [-return_string]
[-max_chains arg] [-quiet] [-verbose]
```

Returns

Report

Usage

Name	Description
[-**file**]	Filename to output results to. (send output to console if -file is not used)
[-**append**]	Append to existing file
[-**return_string**]	return report as string
[-**max_chains**]	Number of chains for which report is to be generated Default: 1
[-**quiet**]	Ignore command errors
[-**verbose**]	Suspend message limits during command execution

Categories

Report

Description

Report the details of the carry chains used by the current open design. The report includes the average depth of all carry chains, as well as the specific depth of each carry chain reported.

By default, the longest carry chain is reported, but the number of chains reported can be specified.

The command returns the carry chain report.

Arguments

-file *arg* - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-return_string - (Optional) 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.

Note This argument cannot be used with the **-file** option.

-max_chains *arg* - (Optional) Number of chains to report. By default the longest carry chain is reported.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example returns the 10 longest carry chains in the design:

```
report_carry_chains -max_chains 10
```

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] [-verbose]
```

Returns

Nothing

Usage

Name	Description
[-**delay_type**]	Type of path delay: Values: max, min, min_max Default: max
[-**setup**]	Consider max delay timing paths (equivalent to -delay_type max)
[-**hold**]	Consider min delay timing paths (equivalent to -delay_type min)
[-**significant_digits**]	Number of digits to display: Range: 0 to 3 Default: 2
[-**file**]	Filename to output results to. (send output to console if -file is not used)
[-**append**]	Append the results to file, don't overwrite the results file
[-**name**]	Output the results to GUI panel with this name
[-**return_string**]	Return report as string
[-**quiet**]	Ignore command errors
[-**verbose**]	Suspend message limits during command execution

Categories

Report

Description

Reports 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 synthesized or implemented design.

Note By default the report is written to the Tcl console or STD output. However, the results can also be written to a file or returned as a string if desired.

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* - The number of significant digits in the output results. The valid range is 0 to 3. The default setting is 2 significant digits.

-file *arg* - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless **-append** is also specified.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-append - Append the output of the command to the specified file rather than overwriting it.

Note The **-append** option can only be used with the **-file** option

-name *arg* - The name of the Clock Interaction Report view to display in the 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 rather than to the standard output. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note This argument cannot be used with the **-file** option.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

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

- report_clocks
- set_delay_model
- set_speed_grade

report_clocks

Report clocks.

Syntax

```
report_clocks [-file arg] [-append] [-return_string] [-quiet]
[-verbose] [clocks]
```

Returns

Nothing

Usage

Name	Description
[**-file**]	Filename to output results to. (send output to console if -file is not used)
[**-append**]	Append the results to file, don't overwrite the results file
[**-return_string**]	return report as string
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*clocks*]	List of clocks Default: *

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 in the current synthesized or implemented design. More detailed information about each clock net can be obtained with the **report_clock_utilization** command.

Note By default the report is written to the Tcl console or STD output. However, the results can also be written to a file or returned as a string if desired.

Arguments

-file *arg* - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless **-append** is also specified.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-append - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note The **-append** option can only be used with the **-file** option

-return_string - (Optional) 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

clocks - (Optional) The clocks to match 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*
```

report_config_timing

Report settings affecting timing analysis.

Syntax

```
report_config_timing [-file arg] [-append] [-name arg]
[-return_string] [-all] [-no_header] [-quiet] [-verbose]
```

Returns

Nothing

Usage

Name	Description
[-**file**]	Output the results to file
[-**append**]	Append the results to file, don't overwrite the results file
[-**name**]	Output the results to GUI panel with this name
[-**return_string**]	return report as string
[-**all**]	report all configuration settings (by default, only the typically important settings are reported
[-**no_header**]	do not generate a report header
[-**quiet**]	Ignore command errors
[-**verbose**]	Suspend message limits during command execution

Categories

Report, Timing

Description

Report the configuration of timing constraints of the current design.

By default the report is abbreviated, containing only a few key timing constraints. Use the **-all** argument to return all timing related configuration.

Arguments

-file *arg* - (Optional) Write the timing constraints configuration report into the specified file. The specified file will be overwritten if one already exists, unless **-append** is also specified.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-append - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note The **-append** option can only be used with the **-file** option

-name *arg* - (Optional) The name of the results to output to the GUI.

-return_string - (Optional) 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.

Note This argument cannot be used with the **-file** option.

-all - (Optional) Reports the state of all timing related attributes and constraints in the design. By default, only a limited set of important timing attributes is reported.

-no_header - (Optional) Disables the report header. By default the report includes a header that lists:

- Report Type - timer_configuration.

- Design - The top module of the design.

- Part - The device, package, and speed grade of the target part.

- Version - The version of software used to create the report

- Date - The date of the report.

- Command - The command options used to create the report.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example reports the current timing configuration, returns the information as a string, and sets that string into the specified Tcl variable:

```
set timeConfig [report_config_timing -all -no_header -return_string]
puts $timeConfig
```

See Also

delete_timing_results

report_control_sets

Report the unique control sets in design.

Syntax

```
report_control_sets [-file arg] [-append] [-sort_by args]
[-cells args] [-return_string] [-quiet] [-verbose]
```

Returns

Report

Usage

Name	Description
[-**file**]	Filename to output results to. (send output to console if -file is not used)
[-**append**]	Append to existing file
[-**sort_by**]	Sort criterion: can be used only when -verbose is used. Options are clk, clkEn, set. Ex: report_control_sets -verbose -sort_by {clk clkEn}
[-**cells**]	Cells/bel_instances for which to report control sets
[-**return_string**]	return report as string
[-**quiet**]	Ignore command errors
[-**verbose**]	Suspend message limits during command execution

Categories

Report

Description

Report the control sets of the current design.

Control sets are the list of control signals (Clock, CE, SR) for SLICE registers and LUTs. Registers must belong to the same control set in order to be packed into the same device resource. Registers without a control signal cannot be packed into devices with registers having control signals. A high number of control sets can cause difficulty fitting the device and can cause routing congestion and timing issues.

By default the **report_control_sets** command returns an abbreviated report indicating only the number of unique control sets. However, the **-verbose** arguments returns a detailed report of all control sets, for either the whole design or for the specified cells.

Arguments

-file *arg* - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-sort_by *args* - (Optional) Sort the detailed report generated by the **-verbose** argument according to the specified criteria. Valid sort criteria are: **clk**, **clkEn**, and **set**.

Note The **-sort_by** argument is used with **-verbose**

-cells *args* - (Optional) Report control sets for the specified cell objects.

-return_string - (Optional) 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.

Note This argument cannot be used with the **-file** option.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - Provide a detailed report listing all control sets of the specified objects. Temporarily override any message limits and return all output from the command.

Examples

The following example reports the control sets of the current design, sorted by the clk and clkEn signals:

```
report_control_sets -verbose -sort_by {clk clkEn}
```

The following example reports the control sets of the specified cells, sorted by clk and set:

```
report_control_sets -verbose -sort_by {clk set} -cells [get_cells usb*]
```

report_datasheet

Report data sheet.

Syntax

```
report_datasheet [-significant_digits arg] [-file arg] [-append]
[-return_string] [-sort_by arg] [-name arg] [-show_all_corners]
[-group args] [-quiet] [-verbose]
```

Returns

Nothing

Usage

Name	Description
[-**significant_digits**]	Number of digits to display: Range: 0 to 3 Default: 3
[-**file**]	Filename to output results to. (send output to console if -file is not used)
[-**append**]	Append the results to file, don't overwrite the results file
[-**return_string**]	return report as string
[-**sort_by**]	Sorting order: Values: clock, port Default: clock
[-**name**]	Output the results to GUI panel with this name
[-**show_all_corners**]	provide all corners
[-**group**]	List of output ports for skew calculation
[-**quiet**]	Ignore command errors
[-**verbose**]	Suspend message limits during command execution

Categories

Report

Description

Create a "datasheet" report for the current design. The datasheet has the timing characteristics of a design at the I/O pads, similar to what is reported in the .twr file.

For example setup and hold times of input I/Os in relation to clocks, max/min delays from clocks to output pads, skews of input/ output buses.

Arguments

-significant_digits *arg* - (Optional) Number of digits to display from 0 to 3. The default is 3.

-file *arg* - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless **-append** is also specified.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-append - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note The **-append** option can only be used with the **-file** option

-return_string - (Optional) 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.

Note This argument cannot be used with the **-file** option.

-sort_by [port | clock] - (Optional) Sorting order. Valid values are clock or port. The default is to sort the report by clocks.

-show_all_corners - (Optional) Report all process corners.

-group [get_ports {xxx1 xxx2 ... xxxN}] - (Optional) Allows you to define your own custom group of ports for analysis. This option requires a list of port objects as returned by the **get_ports** command. The first port in the list will be used as the reference for skew calculation. In most cases, this will be a clock port of a source synchronous output interface. Multiple groups can be specified, each with its own reference clock port. When **-group** is not specified the timer automatically finds the group of output ports based on the launching clock, and reports skew based on that clock.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example returns the datasheet sorted by ports, for all process corners:

```
report_datasheet -sort_by port -show_all_corners
```

The following example reports the datasheet with the skew calculation for two groups of ports, with the first port of each group providing the reference for the skew calculation for that group. In this example, CLK0OUT is the forwarded clock for DATA0-4 and CLK1OUT is forwarded clock for DATA4-7:

```
report_datasheet -file ds.txt -group [get_ports {CLK0OUT DATA0 DATA1 DATA2 DATA3}] \
-group [get_ports {CLK1OUT DATA4 DATA5 DATA6 DATA7}]
```

See Also

get_ports

report_debug_core

Report details on ChipScope debug cores.

Syntax

report_debug_core [**-file** *arg*] [**-append**] [**-return_string**]
[**-quiet**] [**-verbose**]

Returns

Nothing

Usage

Name	Description
[**-file**]	Filename to output results to. (send output to console if -file is not used)
[**-append**]	Append the results to file, don't overwrite the results file
[**-return_string**]	Return report as a string
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

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.

Note By default the report is written to the Tcl console or STD output. However, the results can also be written to a file or returned as a string if desired.

Arguments

-file *arg* - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-append - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note The **-append** option can only be used with the **-file** option

-return_string - Return report as a string. This argument can not be used with the **-file** argument.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

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_default_switching_activity

Get default switching activity on specified default types.

Syntax

```
report_default_switching_activity [-static_probability]
[-toggle_rate] -type args [-file arg] [-return_string] [-append]
[-quiet] [-verbose]
```

Returns

Nothing

Usage

Name	Description
[**-static_probability**]	report static probability
[**-toggle_rate**]	report toggle rate
-type	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
[**-file**]	Filename to output results to. (send output to console if -file is not used)
[**-return_string**]	return default switching activity as string
[**-append**]	append default switching activity to end of file
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Report

Description

Displays the default switching activity currently configured for the specified element type.

The reported values are defined using the **set_default_switching_activity** command.

Note By default the report is written to the Tcl console or STD output. However, the results can also be written to a file or returned as a string if desired.

Arguments

-static_probability - (Optional) Include static probability in the report but do not include toggle rate.

-toggle_rate - (Optional) Include toggle rate in the report but do not include static probability.

Note Both toggle rate and probability will be reported unless either **-toggle_rate** or **-static_probability** is specified to limit the results

-type *<types>* - The component types that are reported. Valid values are: input, input_set, input_reset, input_enable, register, dsp, bram_read_enable, bram_write_enable, output_enable, clock, all.

-file *arg* - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-return_string - (Optional) Return the report as a string rather than a data set.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example reports the default switching attributes for dsps:

```
report_default_switching_activity -type dsp
Default Dsp Probability = 0.50
Default Dsp Toggle Rate (%) = 12.50
```

The following example reports the default switching attributes for all types, and stores it into a Tcl variable swa1:

```
set swa1 [report_default_switching_activity -type all -return_string]
```

Note Without the **-return_string** argument, the command will perform correctly, but the $swa variable will not be assigned the reported information.

See Also

- report_power
- report_switching_activity
- reset_default_switching_activity
- reset_switching_activity
- set_switching_activity

report_drc

Run DRC.

Syntax

report_drc [**-name** *arg*] [**-rules** *args*] [**-ruledeck** *arg*] [**-file** *arg*]
[**-append**] [**-return_string**] [**-quiet**] [**-verbose**]

Returns

Nothing

Usage

Name	Description
[**-name**]	Output the results to GUI panel with this name
[**-rules**]	DRC rules (see get_drc_checks for available rules)
[**-ruledeck**]	Container of DRC rule checks Default: report_drc
[**-file**]	Filename to output results to. (send output to console if -file is not used)
[**-append**]	Append the results to file, do not overwrite the results file
[**-return_string**]	return report as string
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

DRC, Report

Description

Check the current design against a specified set of design rule checks, or a rule deck, and report any errors or violations that are found.

The tool includes a large number of predefined design rule checks to be used by the **report_drc** command. Use the **get_drc_checks** command to list the currently defined design rule checks. You can also create new custom design rule checks using the **create_drc_check** command.

A rule deck is a collection of design rule checks grouped for convenience, to be run at different stages of the FPGA design flow, such as during I/O planning or placement. The tool comes with a set of factory defined rule decks, but you can also create new user-defined rule decks with the **create_drc_ruledeck** command. Use the **get_drc_ruledecks** command to return a list of the currently defined rule decks available for use in the report_drc command.

The **report_drc** command runs a default rule deck when the **-rules** or **-ruledeck** options are not specified. Creating a user-defined DRC automatically adds the new design rule check to the default rule deck.

This command requires an open design to check the design rules against. The command returns a report with the results of violations found by the design rule checks. Violations can be listed with the **get_drc_vios** command.

You can reset the current results of the **report_drc** command, clearing any found violations, using the **reset_drc** command.

Arguments

-name *arg* - (Optional) The name to assign to the results when run in GUI mode.

-ruledeck *arg* - (Optional) The name of a DRC rule deck. A rule deck is a list of DRC rule check names. You can provide the name of a factory DRC rule deck or a user-defined rule deck. The **report_drc** command checks the design against the rules that are added to the given rule deck. Custom rule decks can be defined using the **create_drc_ruledeck** command. Use the **get_drc_ruledecks** command to list the currently defined rule decks.

-rules *args* - (Optional) A list of rules to run the DRC report against. All specified rules will be checked against the current design. Rules are listed by their group name or full key. Using the **-rules** option creates a temporary user-defined rule deck, with the specified design rule checks, and uses the temporary rule deck for the run.

Note **-ruledeck** and **-rules** cannot be used together

-file *arg* - (Optional) Write the DRC report into the specified file. The specified file will be overwritten if one already exists, unless **-append** is also specified.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-append - Append the output of the command to the specified file rather than overwriting it.

Note The **-append** option can only be used with the **-file** option

-return_string - (Optional) 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.

Note This argument cannot be used with the **-file** option.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example lists the available rule decks. The results include all factory rule decks and all user-defined rule decks.

```
get_drc_ruledecks
```

The following example returns the list of DRC rules defined in the specified rule deck:

```
get_drc_checks -of_objects [get_drc_ruledecks placer_checks]
```

The following examples run the specified DRC rule deck and rules against the current design, and writes the results to the specified file:

```
report_drc -ruledeck placer_checks -file C:/Data/DRC_Rpt1.txt
report_drc -rules {IOCNT-1 IOPCPR-1 IOPCMGT-1 IOCTMGT-1 IODIR-1}  \
   -file C:/Data/DRC_Rpt1.txt -append
```

Note The **-append** option adds the result of the second **report_drc** command to the specified file

See Also

- create_drc_check
- create_drc_violation
- reset_drc

report_environment

Report system information.

Syntax

```
report_environment [-file arg] [-append] [-return_string]
[-quiet] [-verbose]
```

Returns

Nothing

Usage

Name	Description
[-file]	Write system information to specified file.
[-append]	Append report to existing file
[-return_string]	Return report content as a string value
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Report

Description

Report the details of the system environment that the tool is running under. The details of the environment report include: operating system version, CPU, memory, available disk space, and specific settings of various environment variables.

The default is to write the report to the standard output. However, the report can be written to a file instead.

Arguments

-file *arg* - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless **-append** is also specified.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-append - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note The **-append** option can only be used with the **-file** option

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example reports the current environment to the specified file:

```
report_environment -file C:/Data/toolEnv.txt
```

report_high_fanout_nets

Report high fanout nets.

Syntax

```
report_high_fanout_nets [-file arg] [-append] [-ascending]
[-load_types] [-clock_regions] [-slr] [-max_nets arg]
[-min_fanout arg] [-max_fanout arg] [-cells args]
[-return_string] [-quiet] [-verbose]
```

Returns

Report

Usage

Name	Description
[-file]	Filename to output results to. (send output to console if -file is not used)
[-append]	Append to existing file
[-ascending]	Report nets in ascending order
[-load_types]	Report load details
[-clock_regions]	Report clock region wise load distribution
[-slr]	Report SLR wise load distribution
[-max_nets]	Number of nets for which report is to be generated Default: 10
[-min_fanout]	Report nets that have fanout greater than or equal to the specified integer Default: 1
[-max_fanout]	Report nets that have fanout less than or equal to the specified integer Default: INT_MAX
[-cells]	Cells/bel_instances for which to report nets
[-return_string]	return report as string
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Report

Description

Report the fanout of nets in the design, starting with the highest fanout nets, and working down. Options allow you to control various aspects of the report.

The command returns the fanout report of nets in the design.

Arguments

-file *arg* - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless **-append** is also specified.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-append - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note The **-append** option can only be used with the **-file** option

-ascending - (Optional) Report nets in ascending order.

-load_types - (Optional) Add details of the loads to the report. This option reports the various load types on the net, after placement.

-clock_regions - (Optional) Report the load distribution across clock regions. This option reports the clock regions the various loads on the net are located in, after placement.

-slr - (Optional) Report the load distribution across SLRs. This option reports the SLRs the various loads on the net are located in, after placement.

-max_nets *arg* - (Optional) Number of nets to report. Default: 10

-min_fanout *arg* - (Optional) Report nets that have fanout greater than or equal to the specified integer. This allows you to report nets with specific fanout loads of interest. Default: 1.

-max_fanout *arg* - (Optional) Report nets that have fanout less than or equal to the specified integer. This allows you to report nets with specific fanout loads of interest. There is no maximum value specified by default.

-cells *args* - (Optional) Cells/bel_instances for which to report nets. Report the nets attached to the specified cells or bels in the design.

-return_string - (Optional) 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.

Note This argument cannot be used with the **-file** option.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example reports the top 100 nets with fanouts greater than 50 loads:

```
report_high_fanout_nets –min_fanout 50 –max_nets 100
```

report_io

Display information about all the IO sites on the device.

Syntax

```
report_io [-file arg] [-append] [-return_string] [-quiet]
[-verbose]
```

Returns

Report

Usage

Name	Description
[-**file**]	Filename to output results to. Send output to console if -file is not used.
[-**append**]	Append to existing file
[-**return_string**]	return report as string
[-**quiet**]	Ignore command errors
[-**verbose**]	Suspend message limits during command execution

Categories

Report

Description

Report details of the IO banks of the current design. Details include device specific information such as target part, package, and speed grade, and also provides information related to each pin on the device.

Arguments

-file *arg* - (Optional) Write the report into the specified file.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-return_string - (Optional) 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.

Note This argument cannot be used with the **-file** option.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example reports the IO blocks of the current design:

```
report_io
```

report_operating_conditions

Get operating conditions values for power estimation.

Syntax

```
report_operating_conditions [-voltage args] [-grade] [-process]
[-junction_temp] [-ambient_temp] [-thetaja] [-thetasa] [-airflow]
[-heatsink] [-thetajb] [-board] [-board_temp] [-board_layers]
[-all] [-file arg] [-return_string] [-append] [-quiet]
[-verbose]
```

Returns

Nothing

Usage

Name	Description
[-voltage]	Gets voltage value. Supported voltage supplies vary by family.
[-grade]	Temperature grade. Supported values vary by family.
[-process]	Gets process
[-junction_temp]	Junction Temperature (C): auto|degC
[-ambient_temp]	Ambient Temperature (C): default|degC
[-thetaja]	ThetaJA (C/W): auto|degC/W
[-thetasa]	Gets ThetaSA
[-airflow]	Airflow (LFM): 0 to 750
[-heatsink]	Gets dimensions of heatsink
[-thetajb]	Gets ThetaJB
[-board]	Board type: jedec, small, medium, large, custom
[-board_temp]	Board Temperature degC
[-board_layers]	Board layers: 4to7, 8to11, 12to15, 16+
[-all]	Gets all operating conditions listed in this help message
[-file]	Filename to output results to. (send output to console if -file is not used)
[-return_string]	return operating conditions as string
[-append]	append operating conditions to end of file
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

[Report](), [SDC]()

Description

Displays the real-world operating conditions that are used when performing analysis of the design.

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. The values of operating conditions can be defined by the **set_operating_conditions** command.

Note By default the report is written to the Tcl console or STD output. However, the results can also be written to a file or returned as a string if desired.

Arguments

-voltage - (Optional) Report the list of voltage pairs. Supported voltage supplies vary by family.

-grade - (Optional) Report the temperature grade of the target device

-process - (Optional) Report the manufacturing process characteristics to be assumed.

-junction_temp - (Optional) Report the device junction temperature used for modeling

-ambient_temp - (Optional) Reports the environment ambient temperature

-thetaja - (Optional) Report the Theta-JA thermal resistance used for modeling

-thetasa - (Optional) Report the Theta-SA thermal resistance used for modeling

-airflow - (Optional) Report the Linear Feet Per Minute (LFM) airflow to be used for modeling.

-heatsink - (Optional) Report the heatsink type to be used for modeling.

-thetajb - (Optional) Report the Theta-JB thermal resistance used for modeling

-board - (Optional) Report the board size to be used for modeling.

-board_temp - (Optional) Report the board temperature in degrees Centigrade to be used for modeling.

-board_layers - (Optional) Report the number of board layers to be used for modeling

-all - (Optional) Report the current values of all operating conditions. Use this to avoid having to report each condition separately.

-file *arg* - (Optional) Write the report into the specified file.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-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.

Note This argument cannot be used with the **-file** option.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

Specify an industrial temperature grade device with an ambient temperature of 75 degrees C and then write those settings to a file on disk.

```
set_operating_conditions -grade industrial -junction_temp 75
report_operating_conditions -grade -junction_temp -return_string -file \
~/conditions.txt
```

See Also

set_operating_conditions

report_param

Get information about all parameters.

Syntax

```
report_param [-file arg] [-append] [-non_default]
[-return_string] [-quiet] [-verbose] [pattern]
```

Returns

Param report

Usage

Name	Description
[**-file**]	Filename to output results to. (send output to console if -file is not used)
[**-append**]	Append the results to file, don't overwrite the results file
[**-non_default**]	Report only params that are set to a non default value
[**-return_string**]	Return report as string
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*pattern*]	Display params matching pattern Default: *

Categories

PropertyAndParameter, Report

Description

Gets a list of all user-definable parameters, the current value, and a description of what the parameter configures or controls.

Arguments

-file *arg* - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless **-append** is also specified.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-append - Append the output of the 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.

Note This argument cannot be used with the **-file** option.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

pattern - Match parameters against the specified pattern. The default pattern is the wildcard '*' which gets all user-definable parameters.

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 [-no_propagation] [-hier arg] [-vid] [-file arg]
[-name arg] [-xpe arg] [-l arg] [-return_string] [-append]
[-fileset arg] [-quiet] [-verbose]
```

Returns

Nothing

Usage

Name	Description
[**-no_propagation**]	Disables the propagation engine to estimate the switching activity of nets.
[**-hier**]	Hierarchy report style (logic, power, or all) Default: logic
[**-vid**]	Voltage ID (VID) of device is used
[**-file**]	Filename to output results to. (send output to console if -file is not used)
[**-name**]	Output the results to GUI panel with this name
[**-xpe**]	Output the results to XML file for importing into XPE
[**-l**]	Maximum number of lines to report in detailed reports (l >= 0) Default: 10
[**-return_string**]	return report as string
[**-append**]	append power report to end of file
[**-fileset**]	Fileset to parse and get .xpe file for PS7 block
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Report, Power

Description

Run power analysis on the current design, and report details of power consumption based on the current operating conditions of the device, and the switching rates of the design. The operating conditions can be set using the **set_operating_conditions** command. The switching activity can be defined using the **set_default_switching_activity** command.

Power analysis requires a synthesized netlist, or a placed and routed design.

Note By default the report is written to the Tcl console or STD output. However, the results can also be written to a file or returned as a string if desired.

Arguments

-no_propagation - (Optional) For all undefined nodes power analysis uses a vector-less propagation engine to estimate activity. This argument disables the propagation engine for a faster analysis of the design.

-hier [logic | power | all] - (Optional) Defines the details (logic, or power) to include in the Detailed Report section of the output. The default is **logic**.

-vid - (Optional) Use the Voltage ID bit of the target device. Voltage identification is a form of adaptive voltage scaling (AVS) that enables certain devices in the Virtex-7 family to be operated at a reduced voltage of 0.9V while delivering the same specified performance of a device operating at the nominal supply voltage of 1.0V. Voltage identification capable devices consume approximately 30% lower worst case (maximum) static power and correspondingly dissipate less heat.

-file *arg* - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless **-append** is also specified.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-append - Append the output of the command to the specified file rather than overwriting it.

Note The **-append** option can only be used with the **-file** option

-name *arg* - (Optional) Specifies the name of the results set to report the results to.

-xpe *arg* - (Optional) Output the results to an XML file for importing into XPower Estimator or XPower Analyzer.

-l *arg* - (Optional) Maximum number of lines to report in the Detailed Reports section. Valid values are greater than or equal to 0.

Note This options also triggers additional levels of detail in the Detailed Report section that are not reported when **-l** is not specified.

-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.

Note This argument cannot be used with the **-file** option.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example performs power analysis, without net propagation, and writes the results to an XML file for use in XPE:

```
report_power –no_propagation –xpe C:/Data/design1.xpe
```

See Also

set_operating_conditions

report_property

Report properties of object.

Syntax

```
report_property [-all] [-return_string] [-file arg] [-append]
[-quiet] [-verbose] object
```

Returns

Property report

Usage

Name	Description
[-all]	Report all properties of object even if not set
[-return_string]	Set the result of running report_property in the Tcl interpreter's result variable
[-file]	Filename to output result to. Send output to console if -file is not used.
[-append]	Append the results to file, don't overwrite the results file
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
object	Object to query for properties

Categories

Object, PropertyAndParameter, Report

Description

Gets the property name, property type, and property value for all of the properties on a specified object.

Note **list_property** also returns a list of properties on an object, but does not include the property type or value.

Arguments

-all - (Optional) Return all properties of an object, even if the property value is not defined.

-return_string - (Optional) Directs the output to a Tcl string. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

-file *arg* - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless **-append** is also specified.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-append - Append the output of the command to the specified file rather than overwriting it.

Note The **-append** option can only be used with the **-file** option

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

object - A single object on which to report properties.

Note If you specify multiple objects you will get an error.

Examples

The following example returns all properties of the specified object:

```
report_property -all [get_cells cpuEngine]
```

To determine which properties are available for the different design objects supported by the tool, you can use multiple **report_property** commands in sequence. The following example returns all properties of the specified current objects:

```
report_property -all [current_project]
report_property -all [current_fileset]
report_property -all [current_design]
report_property -all [current_run]
```

See Also

- create_property
- get_cells
- get_property
- list_property
- list_property_value
- reset_property
- set_property

report_pulse_width

Report pulse width check.

Syntax

```
report_pulse_width [-file arg] [-append] [-name arg]
[-return_string] [-all_violators] [-significant_digits arg]
[-limit arg] [-min_period] [-max_period] [-low_pulse]
[-high_pulse] [-max_skew] [-clocks args] [-quiet] [-verbose]
[objects]
```

Returns

Nothing

Usage

Name	Description
[**-file**]	Filename to output results to. (send output to console if -file is not used)
[**-append**]	Append the results to file, don't overwrite the results file
[**-name**]	Results name in which to store output
[**-return_string**]	return report as string
[**-all_violators**]	Only report pins/ports where check violations occur
[**-significant_digits**]	Number of digits to display: Range: 0 to 3 Default: 2
[**-limit**]	Number of checks of a particular type to report per clock: Default is 1 Default: 1
[**-min_period**]	Only report min period checks
[**-max_period**]	Only report max period checks
[**-low_pulse**]	Only report min low pulse width checks
[**-high_pulse**]	Only report min high pulse width checks
[**-max_skew**]	Only report max skew checks
[**-clocks**]	List of clocks for which to report min pulse width/min period checks
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*objects*]	List of objects to check min pulse width with

Categories

Report

Description

Reports the pulse width of the specified clock signals in the clock network and upon reaching the flip-flop. This command also performs high pulse width checking, using maximum delay for the rising edge and minimum delay for the falling edge of the clock. Performs low pulse width checking using minimum delay for the rising edge, and maximum delay for the falling edge. This results in a worst case analysis for the current Synthesis or Implemented Design because it assumes worst-case delays for both rising and falling edges. This command also reports the maximum skew, or maximum timing separation allowed between clock signals.

The report includes minimum pulse width, maximum pulse width, low pulse width, high pulse width, and max skew checks by default. However, selecting a specific check will disable the other checks unless they are also specified.

The default report is returned to the standard output, but can be redirected to a file, or to a Tcl string variable for further processing. The report is returned to the standard output by default, unless the **-file**, **-return_string**, or **-name** arguments are specified.

Arguments

-file *arg* - (Optional) Write the report into the specified file. If the specified file already exists, it will be overwritten by the new report, unless the **-append** option is also specified.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-append - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note The **-append** option can only be used with the **-file** option

-name *arg* - (Optional) Specifies the name of the results set for the GUI. Pulse Width reports in the GUI can be deleted by the **delete_timing_results** command.

-return_string - (Optional) 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.

Note This argument cannot be used with the **-file** option.

-all_violators - (Optional) Report only the *objects* where violations are found.

-significant_digits *arg* - (Optional) The number of significant digits in the output results. The valid range is 0 to 3. The default setting is 2 significant digits.

-limit *arg* - (Optional) The number of checks of a particular type to report per clock. This is a value >= 1, and the default is 1.

-min_period - (Optional) Report minimum period checks.

-max_period - (Optional) Report maximum period checks.

-low_pulse - (Optional) Report minimum low pulse width checks.

-high_pulse - (Optional) Report minimum high pulse width checks.

-max_skew - (Optional) Check the skew constraints between two clock pins.

Note The default of the **report_pulse_width** command is to report **min_period**, **max_period**, **low_pulse,high_pulse**, and **max_skew**. Specifying one or more of these options configures the command to only report the specified checks.

-clocks *arg* - (Optional) Clocks to report pulse width and period checks. All clocks are checked if the **-clocks** option is not specified.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

objects - (Optional) The pin objects to report the pulse width from. All pins are checked if no *objects* are specified.

Examples

The following example performs the minimum period and low pulse width check, returning the results to a named results set in the GUI:

```
report_pulse_width -min_period -low_pulse -name timing_1
```

See Also

delete_timing_results

report_resources

Run resource estimation and display report.

Syntax

```
report_resources [-hierarchical] [-levels arg] [-file arg]
[-return_string] [-format arg] [-quiet] [-verbose]
```

Returns

Nothing

Usage

Name	Description
[-**hierarchical**]	Report estimates hierarchically
[-**levels**]	Number of levels of hierarchy to be reported: Value >= 1 Default: 1
[-**file**]	Filename to output results (send output to console if -file is not used)
[-**return_string**]	Return report as string
[-**format**]	Format for the resource estimation report: table, xml Default: table
[-**quiet**]	Ignore command errors
[-**verbose**]	Suspend message limits during command execution

Categories

Report

Description

Report the resources consumed by the current open design. The report is returned to the standard output, unless the **-file** or **-return_string** arguments are specified.

Though resource utilization can be reported early in the design process, the report will be more accurate as the design progresses from synthesis through implementation.

Arguments

-hierarchical - (Optional) Report the resources consumed hierarchically in the design under the heading *RTL Hierarchy* in the report. As a default, the **report_resources** command reports resources consumed by the whole design, from the perspective of the top-level of the design.

-levels *arg* - (Optional) Provides a detailed report of resources consumed at each level of the design hierarchy under the heading *Report for Instance <xxx>*. When used with **-hierarchical**, this defines the levels of the design hierarchy to report resources from. The value specified can be >= 1, with the default of 1.

Note This argument only applies when **-hierarchical** is specified. Otherwise this argument is ignored

-file *arg* - (Optional) Write the resource report into the specified file.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-return_string - (Optional) 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.

Note This argument cannot be used with the **-file** option.

-format [table | xml] - (Optional) Specifies the format of the output as either a table or XML. The default output is table.

Note The format applies when **-file** is specified, but is otherwise ignored.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example reports the resources of the design hierarchy to two levels, and writes the results to the specified file:

```
report_resources -hierarchy -levels 2 -file C:/Data/resources_2.txt
```

See Also

report_utilization

report_simlib_info

Report info of simulation libraries.

Syntax

```
report_simlib_info [-file arg] [-append] [-quiet] [-verbose]
[path]
```

Returns

Nothing

Usage

Name	Description
[-**file**]	Output file Default: report_simlib_info.log
[-**append**]	Append mode
[-**quiet**]	Ignore command errors
[-**verbose**]	Suspend message limits during command execution
[*path*]	Report Pre-Compiled library information

Categories

Simulation

report_ssn

Run SSN analysis on the current package and pinout.

Syntax

```
report_ssn [-name arg] [-return_string] [-format arg]
[-file arg] [-append] [-quiet] [-verbose]
```

Returns

Ssn report

Usage

Name	Description
[**-name**]	Output the results to GUI panel with this name
[**-return_string**]	Return report as string
[**-format**]	Report format. Valid arguments are CSV, HTML Default: csv
[**-file**]	Filename to output results to. (send output to console if -file is not used)
[**-append**]	Append the report to the specified file
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

[Report](#)

Description

Perform a simultaneous switching noise (SSN) analysis of the current design. The SSN analysis is an accurate method for predicting how output switching affects interface noise margins. The calculation and estimates are based on a range of variables intended to identify potential noise-related issues in your design and should not be used as final design "sign off" criteria.

SSN analysis provides estimates of the disruption that simultaneously switching outputs can cause on other output ports in the I/O bank, as well as input ports in the case of Spartan-6 devices. The SSN predictor incorporates I/O bank-specific electrical characteristics into the prediction to better model package effects on SSN.

The **report_ssn** command can be affected by the temperature grade of the selected device as defined by the **-grade** option of the **set_operating_condition** command. Setting the temperature grade prior to running noise analysis lets you see how noisy signals can be on Commercial, Extended, Industrial, Q-Grade, or Military grade devices.

By default, **report_ssn** assumes that every port toggles asynchronously. This results in a worst-case noise analysis, which may be overly pessimistic. The **-phase** option lets you consider clocking information available in the design to more accurately report SSN noise. Clocks must be defined using the **create_clock** and **create_generated_clock** commands. The period, phase shift and duty cycle of the generated clocks have significant impact on SSN analysis.

The **report_ssn** command provides a detailed SSN analysis for Spartan-6, Virtex-6, Virtex-7, Kintex-7, and Artix-7 devices. For the Spartan-3, Virtex-4, and Virtex-5 devices, the **report_sso** command provides an approximation of switching noises.

The report is returned to the standard output, unless the **-file**, **-return_string**, or **-name** arguments are specified.

Arguments

-name *arg* - (Optional) Specifies the name of the results to output to the GUI.

-return_string - (Optional) Directs the output to a Tcl string. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note This argument cannot be used with the **-file** option.

-format [CSV | HTML | TXT] - (Optional) Specifies the format of the output as either comma-separated values (CSV), HTML, or an ASCII (TXT) file. The default output is CSV.

Note The format applies when **-file** is specified, but is otherwise ignored.

-file *arg* - (Optional) Write the SSN report into the specified file. The specified file will be overwritten if one already exists, unless **-append** is also specified.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-append - Append the output of the command to the specified file rather than overwriting it.

Note The **-append** option can only be used with the **-file** option

-phase - (Optional) Consider clock switching cycles in SSN analysis to provide a more accurate result.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example performs an SSN analysis on the current design, formats the output as HTML, and writes the output to the specified file:

```
report_ssn -format html -file C:/Data/devSSN.html
```

The following example performs an SSN analysis, with phase analysis, and returns the output to a string which is stored in the specified variable:

```
set devSSN [report_ssn -phase -format html -return_string]
```

Note The **-format** argument in the preceding example is ignored in the absence of **-file**.

See Also

- report_sso
- reset_ssn
- reset_sso

report_sso

Run WASSO analysis on the current package and pinout.

Syntax

```
report_sso -name arg [-return_string] [-file arg]
[-append] [-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] [-verbose]
```

Returns

Sso report

Usage

Name	Description
-name	Output the results to GUI panel with this name
[**-return_string**]	Return report as string
[**-file**]	Filename to output results to. (send output to console if -file is not used)
[**-append**]	Append the report to the specified file
[**-board_thickness**]	Thickness of the PCB in mils
[**-via_diameter**]	Finished via diameter in mils
[**-pad_to_via_breakout_length**]	Pad to via breakout length in mils
[**-breakout_width**]	Breakout width in mils
[**-other_pcb_inductance**]	Other PCB parasitic inductance in nanohenrys
[**-socket_inductance**]	Socket inductance in nanohenrys
[**-ground_bounce**]	Maximum ground bounce in millivolts
[**-output_cap**]	Capacitance per output driver in picofarads
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Report

Description

Performs a weighted average simultaneous switching outputs (WASSO) analysis of the current synthesized or implemented design to validate signal integrity based on the I/O pin and bank assignments in the design.

The **report_sso** command is for Spartan-3, Virtex-4, and Virtex-5 devices, and provides an approximation of switching noises for these devices. For Spartan-6, Virtex-6, Virtex-7, Kintex-7, and Artix-7 devices the tool supports a more detailed SSN analysis using the **report_ssn** command.

The WASSO analysis is performed with the FPGA device in the context of the system-level design. The details of the printed circuit board that the FPGA device will be mounted to is considered as part of the operating environment during analysis. The attributes of the PCB can be defined with the various arguments of the report_sso command, or left with the default values.

Arguments

-name *arg* - Specifies the name of the result set to output.

-return_string - Directs the output to a Tcl string. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note This argument cannot be used with the **-file** option.

-file *arg* - (Optional) Write the SSO report into the specified file. The specified file will be overwritten if one already exists, unless **-append** is also specified.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-append - Append the output of the command to the specified file rather than overwriting it.

Note The **-append** option can only be used with the **-file** option

Specify the following parameters to define the Printed Circuit Board (PCB) that the FPGA is mounted to:

- **-board_thickness** *arg* - (Optional) Specifies the thickness in mils of the PCB that the FPGA device is mounted onto. The default value is 62 mils.
- **-via_diameter** *arg* - (Optional) Specifies the diameter in mils of breakout vias in the PCB. The default value is 12 mils.
- **-pad_to_via_breakout_length** *arg* - (Optional) Specifies the length in mils of the breakout wire (or trace) from the pad on the FPGA package or socket to the breakout via on the PCB. The default value is 33 mils.
- **-breakout_width** *arg* - (Optional) Specifies the width in mils of the breakout wire on the PCB. The default value is 12 mils.
- **-socket_inductance** *arg* - (Optional) Specifies the inductance in nanohenries (nH) of the mounting socket between the FPGA and the PCB if one is used. The default value is 0 nH.
- **-other_pcb_inductance** *arg* - (Optional) Specifies the inductance of the PCB in nanohenries (nH). The default is 0 nH.

Specify the following FPGA device parameters to describe the target device:

- **-ground_bounce** *arg* - (Optional) Specifies the allowed undershoot or overshoot voltage of the signal on the PCB in milliVolts (mV). The lowest permitted value should be used to provide a conservative WASSO analysis. The default value is 600 mW.
- **-output_cap** *arg* - (Optional) Specifies the capacitance in picoFarads (pF) on the output pins of the FPGA device. The default value is 15 pF.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example performs an SSO analysis on the current synthesis or implemented design, using the default settings for board and device, and writes the output to the specified file:

```
report_sso -file C:/Data/devSSO.txt
```

The following example performs an SSO analysis on the current synthesis or implemented design, using the specified values to define the system-level PCB:

```
report_sso -name sso_1 -file C:/Data/mySSO1.txt \
   -board_thickness 55 -via_diameter 9 -pad_to_via_breakout_length 75 \
   -breakout_width 9 -other_pcb_inductance 37 -socket_inductance 19 \
   -ground_bounce 537 -output_cap 12
```

See Also

- report_ssn
- reset_ssn
- reset_sso

report_stats

Report Statistics.

Syntax

```
report_stats [-file arg] [-cell arg] [-pblock arg]
[-clock_region arg] [-format arg] [-level arg] [-all]
[-tables args] [-quiet] [-verbose]
```

Returns

Report

Usage

Name	Description
[-file]	Filename to which results will be written. Writes console if not specified
[-cell]	Write statistics for the specified cell
[-pblock]	Write statistics for the specified Pblock
[-clock_region]	Write statistics for the specified clock region
[-format]	Report format Values: TABLE, CSV, XML Default: TABLE
[-level]	Report level (used with '-cell' or '-pblock') Default: 1
[-all]	Report all levels (used with '-cell' or '-pblock')
[-tables]	List of table types Values: rtlMacro, rtlPrimitive, rtlHierarchy, rtlMemory, rtlPower, rtlPower2, primitive, netBoundary, carryChain, physicalResource, io, RPM, clock, PRModule, PRModule, pblockOverlap, ioBank
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Report

report_switching_activity

Get switching activity on specified objects.

Syntax

```
report_switching_activity [-static_probability] [-signal_rate]
[-file arg] [-return_string] [-append] [-quiet] [-verbose]
[objects...]
```

Returns

Nothing

Usage

Name	Description
[-static_probability]	report static probability
[-signal_rate]	report signal rate
[-file]	Filename to output results to. (send output to console if -file is not used)
[-return_string]	return switching activity as string
[-append]	append switching activity to end of file
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[objects]	objects

Categories

Report

Description

This command is used to report different kinds of switching activity on design nets, ports, pins, and cells. These include simple signal rate and simple static probability on nets, ports, and pins; and state dependent static probabilities on cells.

The reported values are defined using the **set_switching_activity** command.

Note This command returns the switching activity for the specified objects.

Note By default the report is written to the Tcl console or STD output. However, the results can also be written to a file or returned as a string if desired.

Arguments

-static_probability - (Optional) Specifies that the command returns static probability as part of the report.

-signal_rate - (Optional) Specifies that the command returns the signal rate as part of the report.

-file *filename* - (Optional) Write the report to the specified path and file.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-return_string - (Optional) Returns the data as a text string for assignment to a Tcl variable.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

objects - Specifies the list of objects to return switching activity information on.

Examples

The following example reports the signal_rate and static probability value on all output ports in the design:

```
report_switching_activity -signal_rate -static_probability [all_outputs]
```

See Also

- report_default_switching_activity
- report_power
- reset_default_switching_activity
- reset_switching_activity
- set_switching_activity

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*] [**-unique_pins**]
[**-path_type** *arg*] [**-input_pins**] [**-slack_lesser_than** *arg*]
[**-slack_greater_than** *arg*] [**-group** *args*] [**-sort_by** *arg*]
[**-no_report_unconstrained**] [**-user_ignored**] [**-match_style** *arg*]
[**-of_objects** *args*] [**-significant_digits** *arg*] [**-file** *arg*]
[**-append**] [**-name** *arg*] [**-return_string**] [**-quiet**] [**-verbose**]

Returns

Nothing

Usage

Name	Description
[**-from**]	From pins, ports, cells or clocks
[**-rise_from**]	Rising from pins, ports, cells or clocks
[**-fall_from**]	Falling from pins, ports, cells or clocks
[**-to**]	To pins, ports, cells or clocks
[**-rise_to**]	Rising to pins, ports, cells or clocks
[**-fall_to**]	Falling to pins, ports, cells or clocks
[**-through**]	Through pins, ports, cells or nets
[**-rise_through**]	Rising through pins, ports, cells or nets
[**-fall_through**]	Falling through pins, ports, cells or nets
[**-delay_type**]	Type of path delay: Values: max, min, min_max, max_rise, max_fall, min_rise, min_fall Default: max
[**-setup**]	Report max delay timing paths (equivalent to -delay_type max)
[**-hold**]	Report min delay timing paths (equivalent to -delay_type min)
[**-max_paths**]	Maximum number of paths to output when sorted by slack, or per path group when sorted by group: Value >=1 Default: 1
[**-nworst**]	List up to N worst paths to endpoint: Value >=1 Default: 1
[**-unique_pins**]	for each unique set of pins, show at most 1 path per path group
[**-path_type**]	Format for path report: Values: end summary short full full_clock full_clock_expanded Default: full_clock_expanded
[**-input_pins**]	Show input pins in path

Name	Description
[**-slack_lesser_than**]	Display paths with slack less than this Default: 1e+30
[**-slack_greater_than**]	Display paths with slack greater than this Default: -1e+30
[**-group**]	Limit report to paths in this group(s)
[**-sort_by**]	Sorting order of paths: Values: group, slack Default: slack
[**-no_report_unconstrained**]	Do not report infinite slack paths
[**-user_ignored**]	only report paths which have infinite slack because of set_false_path or set_clock_groups timing constraints
[**-match_style**]	Style of pattern matching, valid values are ucf, sdc Default: ucf
[**-of_objects**]	Report timing for these paths
[**-significant_digits**]	Number of digits to display: Range: 0 to 3 Default: 3
[**-file**]	Filename to output results to. (send output to console if -file is not used)
[**-append**]	Append the results to file, don't overwrite the results file
[**-name**]	Output the results to GUI panel with this name
[**-return_string**]	return report as string
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Report, Timing

Description

This command performs timing analysis on the specified timing paths of the current Synthesized or Implemented Design. By default the tool reports the timing path with the worst calculated slack within each path group. However, you can optionally increase the number of paths and delays reported with the use of the **-nworst** or **-max_paths** arguments.

The timing engine runs in "quad" timing mode, analyzing min and max delays for both slow and fast corners. You can configure the type of analysis performed by the **config_timing_corners** command. However, it is not recommended to change the default because this reduces the timing analysis coverage.

Note By default the report is written to the Tcl console or STD output. However, the results can also be written to the GUI, to a file, or returned as a string if desired.

Arguments

-from *args* - (Optional) The starting points of the timing paths to be analyzed. Ports, pins, or cells can be specified as timing path startpoints. You can also specify a clock object, and all startpoints clocked by the named clock will be analyzed.

-rise_from *args* - (Optional) Similar to the **-from** option, but only the rising edge of signals coming from the startpoints are considered for timing analysis. If a clock object is specified, only the paths launched by the rising edge of the clock are considered as startpoints.

-fall_from *args* - (Optional) Similar to the **-from** option, but only the falling edge of signals coming from the startpoints are considered for timing analysis. If a clock object is specified, only the paths launched by the falling edge of the clock are considered as startpoints.

-to *args* - (Optional) The endpoints, or destination objects of timing paths to be analyzed. Ports, pins, and cell objects can be specified as endpoints. A clock object can also be specified, in which case endpoints clocked by the named clock are analyzed.

-rise_to *args* - (Optional) Similar to the **-to** option, but only the rising edge of signals going to the endpoints is considered for timing analysis. If a clock object is specified, only the paths captured by the rising edge of the named clock are considered as endpoints.

-fall_to *args* - (Optional) Similar to the **-to** option, but only the falling edge of signals going to the endpoints is considered for timing analysis. If a clock object is specified, only the paths captured by the falling edge of the named clock are considered as endpoints.

-through *args* - (Optional) Consider only paths through the specified pins, cell instance, or nets during timing analysis. You can specify individual **-through** (or **-rise_through** and **-fall_through**) points in sequence to define a specific path through the design for analysis. The order of the specified through points is important to define a specific path. You can also specify through points with multiple objects, in which case the timing path can pass through any of the specified through objects.

-rise_through *args* - (Optional) Similar to the **-through** option, but timing analysis is only performed on paths with a rising transition at the specified objects.

-fall_through *args* - (Optional) Similar to the **-through** option, but timing analysis is only performed on paths with a falling transition at the specified objects.

-delay_type *arg* - (Optional) Specifies the type of delay to analyze when running the timing report. The valid values are min, max, min_max, max_rise, max_fall, min_rise, min_fall. The default setting for **-delay_type** is max.

-setup - (Optional) Check for setup violations. This is the same as specifying **-delay_type max**.

-hold - (Optional) 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**

-max_paths *arg* - (Optional) The maximum number of paths to output when sorted by slack; or the maximum number of paths per path group when sorted by group, as specified by **-sort_by**. This is specified as a value greater than or equal to 1. By default the **report_timing** command will report the single worst timing path, or the worst path per path group.

-nworst *arg* - (Optional) The number of timing paths to output in the timing report. The timing report will return the *N* worst paths based on the specified value, greater than or equal to 1. The default setting is 1.

-path_type *arg* - (Optional) Specify the path data to output in the timing report. The default format is full_clock_expanded. The valid path types are:

- end - Shows the endpoint of the path only, with calculated timing values.

- summary - Displays the startpoints and endpoints with slack calculation.

- short - Displays the startpoints and endpoints with calculated timing values.

- full - Displays the full timing path, including startpoints, through points, and endpoints.

- full_clock - Displays full clock paths in addition to the full timing path.

- full_clock_expanded - Displays full clock paths between a master clock and generated clocks in addition to the full_clock timing path. This is the default setting.

-input_pins - (Optional) Show input pins in the timing path report. For use with **-path_type** full, full_clock, and full_clock_expanded.

-slack_lesser_than *arg* - (Optional) Report timing on paths with a calculated slack value less than the specified value. Used with **-slack_greater_than** to provide a range of slack values of specific interest.

-slack_greater_than *arg* - (Optional) Report timing on paths with a calculated slack value greater than the specified value. Used with **-slack_lesser_than** to provide a range of slack values of specific interest.

-group *args* - (Optional) Report timing for paths in the specified path groups. Currently defined path groups can be determined with the **get_path_groups** command.

-sort_by [slack | group] - (Optional) Sort timing paths in the report by slack values, or by path group. Valid values are slack or group. By default, the **report_timing** command reports the worst, or **-nworst**, timing paths in the design. However, with **-sort_by group**, the **report_timing** command returns the worst, or **-nworst**, paths of each path group.

Note Each clock creates a path group. Path groups can also be defined with the **group_path** command.

-no_report_unconstrained - (Optional) Do not report timing on unconstrained paths. As a default, **report_timing** will report timing on unconstrained paths.

-match_style [sdc | ucf] - (Optional) Indicates that the patterns for objects matches UCF constraints or SDC constraints. The default is UCF.

-of_objects *args* - (Optional) Report timing on the specified timing paths. Used with the **get_timing_paths** command.

-significant_digits *arg* - (Optional) The number of significant digits in the output results. The valid range is 0 to 3. The default setting is 3 significant digits.

-file *arg* - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless **-append** is also specified.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-append - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note The **-append** option can only be used with the **-file** option

-name *arg* - (Optional) Specifies the name of the results set for the GUI.

-return_string - (Optional) 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.

Note This argument cannot be used with the **-file** option.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example reports the timing for the 5 worst paths in the design, reporting the full timing path, including input pins, with timing values:

```
report_timing -nworst 5 -path_type full -input_pins
```

The following example shows the use of the multiple through points to define both a specific path (through state_reg1) and alternate paths (through count_3 or count_4), and writes the timing results to the specified file:

```
report_timing -from go -through {state_reg1} -through { count_3  count_4 } \
-to done -path_type summary -file C:/Data/timing1.txt
```

See Also

- get_path_groups
- set_msg_limit

report_transformed_primitives

Report details of Unisim primitive transformations.

Syntax

```
report_transformed_primitives [-file arg] [-append]
[-return_string] [-quiet] [-verbose]
```

Returns

Nothing

Usage

Name	Description
[-file]	Output file
[-append]	Append the results to file
[-return_string]	return report as string
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Report

Description

Report the transformed primitives in the current design.

As part of the process of opening the Synthesized design, and loading it into memory, the tool will transform legacy netlist primitives to the supported subset of Unisim primitives.

As a default this report will be written to the standard output. However, the report can also be written to a file or returned to a Tcl string variable for further processing.

Arguments

-file *arg* - (Optional) Write the transformed primitives report into the specified file. The specified file will be overwritten if one already exists, unless **-append** is also specified.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-append - Append the output of the command to the specified file rather than overwriting it.

Note The **-append** option can only be used with the **-file** option

-return_string - (Optional) Directs the output to a Tcl string. The Tcl string can be captured by a variable definition and parsed or otherwise processed.

Note This argument cannot be used with the **-file** option.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example reports the transformed primitives in the current design, and returns the result to the specified Tcl variable:

```
set transPrim [ report_transformed_primitives -return_string ]
```

report_utilization

Compute utilization of device and display report.

Syntax

```
report_utilization [-file arg] [-append] [-pblocks args]
[-cells args] [-return_string] [-packthru] [-name arg]
[-no_primitives] [-omit_locs] [-hierarchical]
[-hierarchical_depth arg] [-quiet] [-verbose]
```

Returns

Report

Usage

Name	Description
[**-file**]	Filename to output results to. (send output to console if -file is not used)
[**-append**]	Append the results to file, don't overwrite the results file
[**-pblocks**]	Report utilization of given list of pblocks
[**-cells**]	Report utilization of given list of cells
[**-return_string**]	Return report as string
[**-packthru**]	Reports LUTs used exclusively as pack-thru
[**-name**]	Output the results to GUI panel with this name
[**-no_primitives**]	Removes "Primitives Section" from report_utilization o/p.
[**-omit_locs**]	Removes "Loced" column from report_utilization o/p.
[**-hierarchical**]	Generates text-based hierarchical report.
[**-hierarchical_depth**]	Specifies the depth level for textual hierachical report Default: 0
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Report

Description

Report the utilization of resources on the target part by the current synthesized or implemented design. The report is returned to the standard output, unless the **-file**, **-return_string**, or **-name** arguments are specified.

Though resource utilization can be reported early in the design process, the report will be more accurate as the design progresses from synthesis through implementation.

Arguments

-file *arg* - (Optional) Write the report into the specified file. The specified file will be overwritten if one already exists, unless **-append** is also specified.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-append - (Optional) Append the output of the command to the specified file rather than overwriting it.

Note The **-append** option can only be used with the **-file** option

-pblocks *arg* - (Optional) Report the resources utilized by one or more Pblocks in the design.

-cells *arg* - (Optional) Report the resources utilized by on or more hierarchical cells in the current design.

-return_string - (Optional) 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.

Note This argument cannot be used with the **-file** option.

-packthru - (Optional) Reports LUTs used for route through purposes. This appears in the utilization report as "LUTs used exclusively as route-thrus".

-name *arg* - (Optional) Specifies the name of the results to output to the GUI.

-no_primitives - (Optional) Remove the Primitives section from the report. The Primitives section reports the number and type of logic primitives used on the device.

-omit_locs - (Optional) Omit the LOCed column from the report. The LOCed column reports the quantity of logic elements that have been placed onto the fabric of the device.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example reports the resources utilized by the Pblocks in the design, and writes the results to the specified file:

```
report_utilization -pblocks [get_pblocks] -file C:/Data/FPGA_Design/pb_util.txt
```

See Also

delete_utilization_results

reset_default_switching_activity

Reset switching activity on default types.

Syntax

```
reset_default_switching_activity [-static_probability]
[-toggle_rate] [-quiet] [-verbose] type...
```

Returns

Nothing

Usage

Name	Description
[-static_probability]	Reset static probability
[-toggle_rate]	Reset toggle rate
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
type	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

Categories

Power

Description

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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

type - 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.

Examples

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_default_switching_activity
- report_power
- report_default_switching_activity
- report_switching_activity
- reset_switching_activity
- set_switching_activity

reset_drc

Remove DRC report.

Syntax

reset_drc [**-name** *arg*] [**-quiet**] [**-verbose**]

Returns

Nothing

Usage

Name	Description
[**-name**]	DRC result name
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

DRC, Report

Description

Clear the DRC results from the specified named result set.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

name - Specifies the name of the DRC results to be cleared. The name is established by the **-name** argument in the **report_drc** command.

Examples

The following example clears the specified results set from memory and the GUI:

```
reset_drc DRC1
```

See Also

report_drc

reset_drc_check

Reset one or more drc checks to factory defaults.

Syntax

reset_drc_check [**-quiet**] [**-verbose**] [*rules...*]

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*rules*]	The list of checks to reset.

Categories

DRC, Object

reset_msg_count

Reset message count.

Syntax

reset_msg_count [**-quiet**] [**-verbose**] *id*

Returns

New message count

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
id	Unique message Id to be reset, e.g "Common 17-99". "reset_msg_count -id *" reset all counters

Categories

Report

Description

Reset the message count for the specified message ID to 0. This restarts the message counter toward the specified message limit. This can be used to reset the count of specific messages that may be reaching the limit, or reset the count of all messages returned by the tool.

Every message delivered by the tool has a unique global message ID that consists of an application sub-system code and a message identifier. This results in a message ID that looks like the following:

```
"Common 17-54"
"Netlist 29-28"
"Synth 8-3295"
```

You can get the current message count for a specific message ID using the **get_msg_count** command.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

-id *arg* - Specifies the message ID to reset the count to 0. Specify * to reset the count of all messages to 0.

Examples

The following example resets the message count for all messages:

```
reset_msg_count -id *
```

See Also

- get_msg_count
- set_msg_limit

reset_msg_limit

Reset message limit.

Syntax

reset_msg_limit [**-severity** *arg*] [**-id** *arg*] [**-quiet**] [**-verbose**]

Returns

New message limit

Usage

Name	Description
[**-severity**]	Message severity to be reset (not valid with -id,) e.g. "ERROR" or "CRITICAL WARNING" Default: ALL
[**-id**]	Unique message Id to be reset (not valid with -severity,) e.g "Common 17-99"
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Report

Description

Restores the default message limit. 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.

The current default limit for all messages returned is 4,294,967,295.

Arguments

-id *value* - The ID of a specific message for which to change the message limit. Each message returned contains its own ID. For instance, "Common 17-54" and "Netlist 29-28".

-severity *value* - 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 {} or "".

- **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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

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 29-28"
```

See Also

- get_msg_limit
- set_msg_limit
- set_msg_severity

reset_msg_severity

Reset Message Severity by ID.

Syntax

reset_msg_severity [**-quiet**] [**-verbose**] *id*

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
id	Unique message id to be set, e.g. ""Common 17-99""

Categories

Report

Description

Restores the specified message ID to its default severity.

Use this command after **set_msg_severity** to restore a specific message ID to its original severity level.

Arguments

id - The ID of the message for which the severity should be reset.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example restores the severity of message ID **common-99** to its original severity:

```
reset_msg_severity common-99
```

The following example restores the severity of message ID **Netlist-1129** to its original severity:

```
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] [-verbose]
```

Returns

Nothing

Usage

Name	Description
[-voltage]	Resets voltage value. Supported voltage supplies vary by family.
[-grade]	Resets temperature grade
[-process]	Resets process
[-junction_temp]	Resets Junction Temperature
[-ambient_temp]	Resets Ambient Temperature
[-thetaja]	Resets ThetaJA
[-thetasa]	Resets ThetaSA
[-airflow]	Resets Airflow
[-heatsink]	Resets dimensions of heatsink
[-thetajb]	Resets ThetaJB
[-board]	Resets Board type
[-board_temp]	Resets Board Temperature
[-board_layers]	Resets Board layers
[-all]	Resets all operating conditions listed in this command line
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

SDC, Power

Description

Resets the specified operating conditions to their default values. 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.

Operating conditions can be set using the **set_operating_conditions** command. The current values can be determined using the **report_operating_conditions** command.

You must specify the arguments to reset, or no values are reset. To reset all operating conditions, use the **-all** option.

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

Arguments

-voltage *args* - (Optional) Reset the voltage supply to the default value. The voltage supply and its default depend on the device family.

-grade - (Optional)) Reset the temperature 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 temperature for the target device. The default value is "auto".

-ambient_temp - (Optional) Reset the ambient temperature 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 "12to15".

-all - (Optional) Reset all operating conditions to their default value. Use this to avoid having to reset each condition separately.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

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
```

The following example resets all the operating conditions for the design to their default setting:

```
reset_operating_conditions -all
```

The following example resets the voltage supply Vccint to its default value:

```
reset_operating_conditions -voltage Vccint
```

See Also

- report_operating_conditions
- report_power
- set_operating_conditions

reset_param

Reset a parameter.

Syntax

reset_param [**-quiet**] [**-verbose**] *name*

Returns

Original value

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
name	Parameter name

Categories

PropertyAndParameter

Description

Restores a user-definable configuration parameter that has been changed with the **set_param** command to its default value.

You can use the **report_param** command to see which parameters are currently defined.

Arguments

name - The name of a parameter to reset. You can only reset one parameter at a time.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example restores the tcl.statsThreshold parameter to its default value:

```
reset_param tcl.statsThreshold
```

See Also

- get_param
- list_param
- report_param
- set_param

reset_project

Reset current project.

Syntax

```
reset_project [-exclude_runs] [-exclude_ips] [-exclude_sim_runs]
[-quiet] [-verbose]
```

Returns

Nothing

Usage

Name	Description
[**-exclude_runs**]	Do not reset runs
[**-exclude_ips**]	Do not reset ips
[**-exclude_sim_runs**]	Do not reset simulation runs
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Project

Description

Reset the current project to its starting condition, with source and constraint files, by cleaning out the various output files created during synthesis, simulation, implementation, and write_bitstream. Also resets the state of the project to the start of the design flow.

Arguments

-exclude_runs - (Optional) Exclude the *<project>*.runs folder from the reset process. In this case, the runs folder will be preserved, while the rest of the project data will be removed.

-exclude_ips - (Optional) Exclude the *<project>*.srcs/sources_1/ip folder from the reset process. In this case, the IP folder, containing the IP cores and generated targets, will be preserved, while the rest of the project data will be removed.

-exclude_sim_runs - (Optional) Exclude the *<project>*.sim folder from the reset process. In this case, the simulation folder will be preserved, while the rest of the project data will be removed.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Example

The following example resets the current project, while preserving the simulation run data, and returning all messages regardless of message limits:

```
reset_project -exclude_sim_runs -verbose
```

See Also

- create_project
- current_project

reset_property

Reset property on object(s).

Syntax

reset_property [**-quiet**] [**-verbose**] *property_name objects...*

Returns

The value that was set if success, "" if failure

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
property_name	Name of property to reset
objects	Objects to set properties

Categories

Object, PropertyAndParameter

Description

Restores 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

property_name - The name of the property to be reset.

objects - One or more objects on which the property will be restored to its default value.

Examples

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 [-prev_step] [-from_step arg] [-noclean_dir] [-quiet]
[-verbose] run
```

Returns

Nothing

Usage

Name	Description
[-prev_step]	Reset last run step
[-from_step]	First Step to reset
[-noclean_dir]	Do not remove all output files and directories from disk
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
run	Run to modify

Categories

Project

Description

Resets 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 values for ISE implementation are: NGDBuild, MAP, PAR, TRCE, XDL.

-noclean_dir - Do not clean up the run files output to the run directory. As a 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

run - The name of 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_simulation

Reset an existing simulation run.

Syntax

```
reset_simulation [-mode arg] [-type arg] [-quiet] [-verbose]
[simset]
```

Returns

Nothing

Usage

Name	Description
[**-mode**]	Remove generated data for the specified mode. Values: behavioral, post-synthesis, post-implementation Default: behavioral
[**-type**]	Remove generated data for the specified type. Applicable mode is post-synthesis or post-implementation. Values: functional, timing
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*simset*]	Name of the simulation fileset to reset

Categories

Simulation

reset_ssn

Clear a SSN results set from memory.

Syntax

reset_ssn [**-quiet**] [**-verbose**] *name*

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
name	Name of the set of results

Categories

[Report](#)

Description

Clear the SSN results from the specified named result set.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

name - Specifies the name of the results to be cleared.

Examples

The following example clears the specified results set from memory:

```
reset_ssn SSN1
```

See Also

- [report_ssn](#)
- [report_sso](#)
- [reset_sso](#)

reset_sso

Clear a WASSO results set from memory.

Syntax

reset_sso [**-quiet**] [**-verbose**] *name*

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
name	Name of the set of results

Categories

Report

Description

Clear the specified SSO results from the named result set.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

name - Specifies the name of the results to be cleared.

Examples

The following example clears the specified results set from memory:

```
reset_sso SSO1
```

See Also

- report_ssn
- report_sso
- reset_ssn

reset_switching_activity

Reset switching activity on specified objects.

Syntax

```
reset_switching_activity [-static_probability] [-signal_rate]
[-hier] [-quiet] [-verbose] objects...
```

Returns

Nothing

Usage

Name	Description
[**-static_probability**]	Reset static probability
[**-signal_rate**]	Reset signal rate
[**-hier**]	Hierarchically resets the switching activity on a hierarchical instance provided via -objects option. This option should be used only with -objects option
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
objects	Objects to reset switching activity on

Categories

[Power](#)

Description

Resets the attributes of the 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.

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.

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.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

objects - The list of objects for which to reset the switching activity.

Examples

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_default_switching_activity
- report_power
- report_switching_activity
- reset_default_switching_activity
- set_switching_activity

reset_target

Reset target data for the specified source.

Syntax

reset_target [**-quiet**] [**-verbose**] *name objects*

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
name	List of targets to be reset, or 'all' to reset all generated targets
objects	The objects for which data needs to be reset

Categories

Project, XPS, IPFlow

Description

Remove the current target data for the specified IP core. This deletes any files that were delivered during generation of the specified targets. This does not remove the core from the current project, but does remove the associated target data from its referenced location.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

name - Specifies the name of the type of target to reset. Valid values are:

- **all** - Reset all targets for the specified core.
- **synthesis** - Reset the synthesis netlist for the specified core. This will remove the netlist files for the specified core.
- **instantiation_template** - Reset the instantiation template for the specified core.

objects - The IP core objects to remove the target data from.

Examples

The following example resets the instantiation template for the specified IP core:

```
reset_target instantiation_template [get_ips blk_mem*]
```

See Also

generate_target

reset_timing

Resets the timing information on the current design.

Syntax

reset_timing [**-invalid**] [**-quiet**] [**-verbose**]

Returns

Nothing

Usage

Name	Description
[**-invalid**]	Also rest invalid timing constraints.
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Report, Timing

Description

Reset the timing data for the current design. Use this command to clear the current in-memory timing data, and force the timing engine to reevaluate the design comprehensively rather than iteratively.

Note This command deletes the in-memory timing view, not the timing report. Use the **delete_timing_results** command to delete the reported timing information.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example clears the current timing data from memory:

```
reset_timing
```

See Also

- delete_timing_results
- report_timing

reset_ucf

Clear floorplan constraints read in from a file.

Syntax

reset_ucf [**-quiet**] [**-verbose**] *file*

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
file	UCF file to be reset

Categories

Floorplan

Description

Clear placement constraints defined in the specified UCF constraints file from the current design. This command requires an open design.

The constraints found in the specified file will be removed from the current design, but are not immediately saved to the constraints file. The specified constraint file will be updated when you use the **save_design** command to rewrite the constraints.

You can save the constraints to a new file without saving the design by using the **write_ucf** command.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

file - The UCF file to be reset.

Note The tool will look for the specified file in the constraint filesets

Examples

The following example removes placement constraints found in the specified file:

```
reset_ucf top_full.ucf
```

See Also

write_ucf

resize_net_bus

Resize net bus in the current design.

Syntax

```
resize_net_bus [-from arg] [-to arg] [-quiet]
[-verbose] net_bus_name ...
```

Returns

Nothing

Usage

Name	Description
[**-from**]	New starting bus index
[**-to**]	New ending bus index
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
net_bus_name	Name of the net bus to resize

Categories

Netlist

Description

Resize an existing bus net, to grow the bus, shrink the bus, or renumber the current range of indexes. You can only do a single grow, shrink, or renumber operation with each command.

- You can grow the bus by indicating a new range of indexes outside the current range of indexes. Growing the bus leaves existing bits connected as they currently are.

- You can shrink the bus by indicating a new range of indexes inside the current range of indexes. Shrinking the bus, eliminates connections to removed bits, but leaves the remaining bits connected as they currently are.

- You can renumber the current bus indexes by providing a new range of indexes with the same width as the current range. Renumbering bits changes bus bit numeric identifiers, but doesn't otherwise change connections.

Netlist editing changes the in-memory view of the netlist in the current design. It does not change the files in the source fileset, or change the persistent design on the disk. Changes made to the netlist may be saved to a design checkpoint using the **write_checkpoint** command, or may be exported to a netlist file such as Verilog, VHDL, or EDIF, using the appropriate **write_*** command.

Note Netlist editing is not allowed on the elaborated RTL design.

This command returns nothing if successful, and returns an error if it fails.

Arguments

-from *arg* - (Optional) The new starting index of the specified bus net.

-to *arg* - (Optional) The new ending index of the specified bus.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

net_bus_name - (Required) The names of an existing bus net.

Example

The following example creates a new 24-bit bus, then renumbers the bus indexes to include negative indexes, and then resizes the bus to shrink it to an 8-bit bus:

```
create_net tempBus -from 23 -to 0
resize_net_bus tempBus -from -12 -to 11
resize_net_bus tempBus -from 0 -to 7
```

See Also

- create_pin
- create_port
- get_nets
- resize_pin_bus
- resize_port_bus
- write_edif
- write_verilog
- write_vhdl

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**] [**-verbose**] *pblock*

Returns

Nothing

Usage

Name	Description
[**-add**]	Add site ranges(s)
[**-remove**]	Remove site ranges(s)
[**-from**]	Site range(s) to move
[**-to**]	Site range destination(s)
[**-replace**]	Remove all existing ranges
[**-locs**]	LOC treatment Default: keep_all
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
pblock	Pblock to resize

Categories

Floorplan, XDC

Description

Place, resize, move, or remove the specified Pblock. The Pblock must have been created using the **create_pblock** command.

A Pblock consists of a group of cells that can be assigned to one or more independent or overlapping rectangles. Using the various options defined below, you can add sites to a rectangle, or remove sites from a rectangle, or define a new rectangle to be associated with an existing Pblock.

Arguments

-add *args* - Add the specified range of sites 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 Multiple site types are added as separate rectangles.

-remove *args* - Remove the specified range of sites from the Pblock. Removing sites 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 site or range of sites to relocate from one location to another.

-to *args* - The **-from** and **-to** options must be used as a pair, and specify a site or range of sites to relocate from one location to another.

-locs *args* - Specifies how the placed logic in the Pblock will be handled as the Pblock is moved or resized. Valid 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 - Remove all rectangles associated with the Pblock.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

pblock - Specify the Pblock to be resized, moved, or removed.

Examples

The following example resizes the Pblock by adding a range of SLICEs, and removing other SLICEs, but keeps all instances placed at their current location:

```
resize_pblock block3 -add SLICE_X6Y67:SLICE_X11Y71 -remove SLICE_X6Y71:SLICE_X7Y71 \
-locs keep_all
```

The following example moves the specified Pblock by adding a range of SLICEs, removing the existing range of SLICEs, and trims any placed logic that falls outside the new Pblock. Then it adds a new range of SLICEs and block ram 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 RAMB18_X0Y2:RAMB18_X1Y4}
```

See Also

- add_cells_to_pblock
- create_pblock
- place_pblocks

resize_pin_bus

Resize pin bus in the current design.

Syntax

```
resize_pin_bus [-from arg] [-to arg] [-quiet]
[-verbose] pin_bus_name ...
```

Returns

Nothing

Usage

Name	Description
[-**from**]	New starting bus index
[-**to**]	New ending bus index
[-**quiet**]	Ignore command errors
[-**verbose**]	Suspend message limits during command execution
pin_bus_name	Name of the pin bus to resize

Categories

Netlist

Description

Resize an existing bus pin, to grow the bus, shrink the bus, or renumber the current range of pin indexes. You can only do a single grow, shrink, or renumber operation with each command.

- You can grow the bus by indicating a new range of pin indexes outside the current range of indexes. Growing the bus leaves existing pins connected as they currently are.

- You can shrink the bus by indicating a new range of pin indexes inside the current range of indexes. Shrinking the bus, eliminates connections to removed bus pins, but leaves the remaining pins connected as they currently are.

- You can renumber the current bus indexes by providing a new range of pin indexes with the same width as the current range. Renumbering pins changes the pin index, but does not otherwise change connections.

Netlist editing changes the in-memory view of the netlist in the current design. It does not change the files in the source fileset, or change the persistent design on the disk. Changes made to the netlist may be saved to a design checkpoint using the **write_checkpoint** command, or may be exported to a netlist file such as Verilog, VHDL, or EDIF, using the appropriate **write_*** command.

Note Netlist editing is not allowed on the elaborated RTL design.

This command returns nothing if successful, and returns an error if it fails.

Arguments

-from *arg* - (Optional) The new starting index of the specified bus pin.

-to *arg* - (Optional) The new ending index of the specified bus pin.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

bus_pin_name - (Required) The name of the bus pin to modify. You must specify the pin names hierarchically from the cell instance the pin is assigned to. Pins created at the top-level of the design are ports, and should be resized with the **resize_port_bus** command.

Examples

The following example creates a blackbox cell, then creates a 24-bit bidirectional bus for the specified hierarchical cell, then resizes the bus pin to expand the width to 32-bits, then renumbers the index to include negative bus indexes:

```
create_cell -reference dmaBlock -black_box usbEngine0/myDMA
create_pin -direction INOUT -from 0 -to 23 usbEngine0/myDMA/dataBus
resize_pin_bus -from 0 -to 31 usbEngine0/myDMA/dataBus
resize_pin_bus -from -16 -to 15 usbEngine0/myDMA/dataBus
```

See Also

- create_pin
- create_port
- get_pins
- remove_pin
- resize_net_bus
- resize_port_bus
- write_edif
- write_verilog
- write_vhdl

resize_port_bus

Resize port bus in the current design.

Syntax

```
resize_port_bus [-from arg] [-to arg] [-quiet]
[-verbose] port_bus_name ...
```

Returns

Nothing

Usage

Name	Description
[**-from**]	New starting bus index
[**-to**]	New ending bus index
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
port_bus_name	Name of the port bus to resize

Categories

PinPlanning

Description

Resize an existing bus port, to grow the bus, shrink the bus, or renumber the current range of port indexes. You can only do a single grow, shrink, or renumber operation with each command.

- You can grow the bus by indicating a new range of port indexes outside the current range of indexes. Growing the bus leaves existing port indexes connected as they currently are.

- You can shrink the bus by indicating a new range of port indexes inside the current range of indexes. Shrinking the bus, eliminates connections to removed bus ports, but leaves the remaining ports connected as they currently are.

- You can renumber the current bus indexes by providing a new range of port indexes with the same width as the current range. Renumbering ports changes the port index, but does not otherwise change connections.

Netlist editing changes the in-memory view of the netlist in the current design. It does not change the files in the source fileset, or change the persistent design on the disk. Changes made to the netlist may be saved to a design checkpoint using the **write_checkpoint** command, or may be exported to a netlist file such as Verilog, VHDL, or EDIF, using the appropriate **write_*** command.

Note Netlist editing is not allowed on the elaborated RTL design.

This command returns nothing if successful, and returns an error if it fails.

Arguments

-from *arg* - (Optional) The new starting index of the specified bus port.

-to *arg* - (Optional) The new ending index of the specified bus port.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

bus_port_name - (Required) The name of the bus port to modify.

Examples

The following example creates a 32-bit output bus port, then renumbers the ports to include negative bus indexes, then shrinks the bus width from 32-bits to 16-bits:

```
create_port -direction out -from 0 -to 31 outPorts
resize_port_bus -from -16 -to 15 outPorts
resize_port_bus -from -8 -to 7 outPorts
```

See Also

- create_pin
- create_port
- get_ports
- remove_port
- resize_net_bus
- resize_pin_bus
- write_edif
- write_verilog
- write_vhdl

save_constraints

Save the current constraints.

Syntax

```
save_constraints [-force] [-quiet] [-verbose]
```

Returns

Nothing

Usage

Name	Description
[**-force**]	Force constraints save, overwriting the target XDC if necessary
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Project

Description

Saves any changes to the constraints files of the active constraints set. This command writes any changes to the constraints files to the project data on the hard drive; saving any work in progress and committing any changes.

Arguments

-force - Save the active constraints files regardless of whether any changes have been made, overwriting the current target constraints file.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example saves the constraints files for the active constraints set regardless of any changes to the files:

```
save_constraints -force
```

See Also

save_constraints_as

save_constraints_as

Save current constraints as a new set of constraints.

Syntax

save_constraints_as [**-dir** *arg*] [**-target_constrs_file** *arg*]
[**-quiet**] [**-verbose**] *name*

Returns

Nothing

Usage

Name	Description
[**-dir**]	Directory to save constraints to
[**-target_constrs_file**]	Target constraints file for the new fileset. If a path is not specified, the file will be found or created in the fileset directory.
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
name	Name of new constraints fileset

Categories

Project

Description

Copies the active constraints set to create a new constraints set, with local copies of any constraints files that are part of the constraints set. You can also specify a new constraints file to use as the target for the copied constraints set.

Use this command to save changes to the constraints in a design without affecting the current constraints files. This allows you to do some "what-if" type development of design constraints.

Note The new constraint set created by the **save_constraints_as** command will not be active in the design, although it will be referenced by the design. To make the constraints set active you must set the constrset property to point to the new constraints set for specific runs. See the example below.

Arguments

-dir *arg* - The directory into which constraints files are saved. If the directory is not specified, the new constraints set is located in the project sources directory. The constraints files from the active constraints set are copied into the specified directory.

-target_constrs_file *arg* - Specifies a new target constraints file for the new constraints fileset. If a path is not specified as part of the file name, the file will be created in the fileset directory.

Note You must specify the **.xdc** file extension, or the command will report a warning that the filetype is invalid, and cannot be set to the target constraint set. In this case, the existing target constraints file will be used as the target

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

name - Specifies the name of the constraints set to write.

Examples

The following example saves the active constraints set into a new constraints set called constrs_2, and copies any constraints files into the specified directory, as well as creating a new target constraints file for the constraints set:

```
save_constraints_as -dir C:/Data/con1 -target_constrs_file rev1.xdc constrs_2
```

The following example saves the active constraints set as a new constraints 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 constraints set:

```
save_constraints_as newCon2
set_property CONSTRSET newCon2 [get_runs synth_1]
set_property CONSTRSET newCon2 [get_runs impl_1]
```

Note The constraints set is not active in the design until it has been set to active for the current runs.

See Also

save_constraints

save_project_as

Save the current project under a new name.

Syntax

save_project_as [**-force**] [**-quiet**] [**-verbose**] *name* [*dir*]

Returns

Saved project object

Usage

Name	Description
[**-force**]	Overwrite existing project directory
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
name	New name for the project to save
[*dir*]	Directory where the project file is saved Default: .

Categories

Project

Description

Saves a currently open project file under a new name in the specified directory, or in the current working directory if no other directory is specified.

Arguments

-force - Overwrite the existing project. If the project name is already define in the specified directory then you must also specify the **-force** option for the tool to overwrite the existing project.

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

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

name - The name of the new project. This argument must appear before the specified directory. Since these commands do not have parameters, the 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 - The directory name in which to write the new project file. If the specified directory does not exist a new directory will be created. If the directory is specified with the complete path, the tool uses the specified path name. However, if *dir* is specified without a path, the tool looks for or creates the directory in the current working directory, or the directory from which the tool was launched.

Note When creating a project in GUI-mode, the tool appends the filename *name* to the directory name *dir*I 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 tool will create the project in the current working directory, or the directory from which the 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 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 [**-add**] [**-quiet**] [**-verbose**] *objects*

Returns

Nothing

Usage

Name	Description
[**-add**]	Add to existing selection list
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
objects	Objects to select

Categories

GUIControl

Description

Selects the specified object in the appropriate open views in the 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

objects - Specifies one or more objects to be selected.

Examples

The following example selects the specified site on the device:

```
select_objects [get_sites SLICE_X56Y214]
```

See Also

- get_selected_objects
- unselect_objects

set_delay_model

Timing Delay Model.

Syntax

set_delay_model [**-interconnect** *arg*] [**-quiet**] [**-verbose**]

Returns

Nothing

Usage

Name	Description
[**-interconnect**]	Interconnect delay model used for timing analysis: Values: estimated, actual(default), none
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Timing

Description

Sets the interconnect delay model for timing analysis. There are three settings for the interconnect delay model: "actual", "estimated", or "none".

- If "actual" is selected, the actual delay from the routed interconnect will be used in timing analysis. If the design is only partially routed, then the actual delay from the routed portion will be used, along with estimated delay for the unrouted portion. The timing report will provide details regarding the source of the calculated delay.

- 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. Estimated delay can be specified even if the design is fully routed.

- If "none" is selected, then no interconnect delay is included in the timing analysis, and only the logic delay is applied.

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

Arguments

-interconnect [actual | estimated | none] - (Optional) Delay model to be used. The default setting is **actual**.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following command will use a timing delay model which is an estimated value.

```
set_delay_model -interconnect estimated
```

See Also

report_timing

set_external_delay

Set external delay.

Syntax

```
set_external_delay -from arg -to arg [-min] [-max] [-quiet]
[-verbose] delay_value
```

Returns

Nothing

Usage

Name	Description
-from	Output port
-to	Input port
[**-min**]	Specifies minimum delay
[**-max**]	Specifies maximum delay
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
delay_value	External (feedback) delay value

Categories

XDC

Description

Sets the external (feedback) delay between an output and input port. The external delay is used in the calculation of the PLL/MMCM compensation delay for PLLs/MMCMs with external feedback.

A min or max value can be specified. By default the value specified applies to both min (hold) and max (setup) compensation delays.

The command returns the defined delay.

Arguments

-from *arg* - The output port name.

-to *arg* - The input port name.

-min - (Optional) Specifies the delay_value is a minimum delay value for hold time analysis.

-max - (Optional) Specifies the delay_value is a maximum delay value for setup analysis.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

delay_value - The external delay value. The default value is 0.

Examples

The following example sets the external feedback delay to 1.0 ns between the port ClkOut and ClkFb:

```
set_external_delay -from [get_ports ClkOut] -to [get_ports ClkFb] 1.0
```

See Also

report_timing

set_hierarchy_separator

Set hierarchical separator character.

Syntax

set_hierarchy_separator [**-quiet**] [**-verbose**] [*separator*]

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*separator*]	Hierarchy separator character Default: /

Categories

SDC, XDC

Description

Sets 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 - (Optional) The new character to use as a hierarchy separator. Valid characters to use as the hierarchy separator are: '/', '@', '^', '#', '.', and '|'. The default character is '/', and is used when no *separator* is specified.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

This example changes the hierarchy separator to the '|' character:

```
set_hierarchy_separator |
```

The following example restores the default hierarchy separator, '/':

```
set_hierarchy_separator
```

See Also

get_hierarchy_separator

set_logic_unconnected

Sets logic dc for port/pins.

Syntax

set_logic_unconnected [**-quiet**] [**-verbose**] *objects*

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
objects	List of ports or pins

Categories

SDC, XDC

Description

Defines the specified output ports or pins as unconnected.

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

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

objects - A list of the output ports and pins to be affected.

Examples

The following example sets the specified port to unconnected:

```
set_logic_unconnected [get_ports OUT1]
```

set_msg_limit

Set message limit.

Syntax

```
set_msg_limit [-severity arg] [-id arg] [-quiet]
[-verbose] count
```

Returns

New message limit

Usage

Name	Description
[-severity]	Message severity to be set (not valid with -id,) e.g. "ERROR" or "CRITICAL WARNING" Default: ALL
[-id]	Unique message id to be set (not valid with -severity,) e.g "Common 17-99"
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
count	New message limit

Categories

Report

Description

Defines the number of messages that will be returned by the tool during a design session, or single invocation. You can specify the limit of a specific message ID, or the limit for a specific severity of messages, or the limit for the total number of messages returned by the tool.

Every message delivered by the tool has a unique global message ID that consists of an application sub-system code and a message identifier. This results in a message ID that looks like the following:

```
"Common 17-54"
"Netlist 29-28"
"Synth 8-3295"
```

You can specify the limit for a specific message ID, such as "Common 17-54", or specify a limit for all messages of a specific severity, such as "CRITICAL WARNING". When a message ID or message severity has not been specified, this command will set the limit for the total number of messages returned by the tool.

You can report the current message limit of a message severity or ID with the **get_msg_limit** command. You can restore the default message limit using the **reset_msg_limit** command.

The default message limit for all message IDs is defined by the parameter **messaging.defaultLimit**. You can report the current value of this parameter with the **get_param** command, and change it as needed using the **set_param** command.

Note You can change the severity of a specific message ID with the **set_msg_severity** command

Arguments

-id *value* - The message ID, which is included in all returned messages. For example,"Common 17-54" and "Netlist 29-28".

-severity *value* - 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

count - The message limit specified as a value >= 1. You can restore the message limit to the **messaging.defaultLimit** by specifying a count of -1.

Examples

The following example sets the message limit of the specified message ID:

```
set_msg_limit -id "Netlist 29-28" 10000
```

The following example sets the limit of the specified message severity:

```
set_msg_limit -severity WARNING 50000
```

The following example sets the limit of all messages returned by the tool as specified:

```
set_msg_limit 500000
```

The following example gets the current default message limit, specifies a new default value, then overrides the default for the specified message id:

```
get_param messaging.defaultLimit
    50
set_param messaging.defaultLimit 1000
set_msg_limit -id common-99 1500
```

The following example restores the default message limit:

```
set_msg_limit -1
reset_msg_limit
```

Note Both lines restore the default message limit.

See Also

- get_msg_limit
- get_param
- reset_msg_limit
- set_msg_severity
- set_param

set_msg_severity

Set Message Severity by ID.

Syntax

set_msg_severity [**-quiet**] [**-verbose**] *id severity*

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
id	Unique message id to be set, e.g. "Common 17-54"
severity	Message severity to be changed to, e.g. "ERROR" or "CRITICAL WARNING"

Categories

Report

Description

Sets the severity of a specified message ID from one type, such as WARNING, to another type, such as ERROR.

Every message delivered by the tool has a unique global message ID that consists of an application sub-system code and a message identifier. This results in a message ID that looks like the following:

```
"Common 17-54"
"Netlist 29-28"
"Synth 8-3295"
```

Use this command to customize the message severity returned by the tool to specific levels appropriate to your usage.

Note You can restore the message severity of a specific message ID to its original setting with the **reset_msg_severity** command.

Arguments

id - The message ID, which is included in all returned messages. For example, "Common 17-54" or "Netlist 29-28".

severity - 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 {} or "".

- **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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example reduces the significance of message ID **Common 17-54** from a CRITICAL WARNING to a WARNING so that it causes less concern when encountered:

```
set_msg_severity "Common 17-54" WARNING
```

The following example elevates a common INFO message to a Critical Warning:

```
set_msg_severity "Common 17-81" "CRITICAL WARNING"
```

See Also

- get_msg_limit
- reset_msg_severity
- set_msg_limit

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] [-verbose]
```

Returns

Nothing

Usage

Name	Description
[**-voltage**]	List of voltage pairs, e.g., {name value}. Supported voltage supplies vary by family.
[**-grade**]	Temperature grade. Supported values vary by family. Default: commercial
[**-process**]	Process data: typical or maximum Default: typical
[**-junction_temp**]	Junction Temperature (C): auto|degC Default: auto
[**-ambient_temp**]	Ambient Temperature (C): default|degC Default: default
[**-thetaja**]	ThetaJA (C/W): auto|degC/W Default: auto
[**-thetasa**]	ThetaSA (C/W): auto|degC/W Default: auto
[**-airflow**]	Airflow (LFM): 0 to 750 Default: varies by family
[**-heatsink**]	Dimensions of heatsink: none, low, medium, high, custom Default: medium
[**-thetajb**]	ThetaJB (C/W): auto|degC/W Default: auto
[**-board**]	Board type: jedec, small, medium, large, custom Default: medium
[**-board_temp**]	Board Temperature degC
[**-board_layers**]	Board layers: 4to7, 8to11, 12to15, 16+ Default: 8to11
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

SDC, XDC, Power

Description

Sets the real-world operating conditions that are used when performing analysis of the design. 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.

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

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.

Arguments

-voltage *arg* - (Optional) List of voltage supply names and their values specified in pairs. Supported voltage supply names and their values vary by family. For example if a family supports a voltage supply named Vccint, you can set the supply voltage to 0.8 with the following argument and value : **-voltage {Vccint 0.8}**

-grade *arg* - (Optional) The temperature grade of the target device. Supported values vary by family. The default value is "commercial".

-process *arg* - (Optional) The manufacturing process characteristics to assume. Valid values are "typical" or "maximum". The default value is "typical".

-junction_temp *arg* - (Optional) 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) The environment ambient temperature in degrees C. The default setting is "default".

-thetaja *arg* - (Optional) The Theta-JA thermal resistance used for modeling in degrees C/W. The default setting is "auto".

-thetasa *arg* - (Optional) The Theta-SA thermal resistance used during modeling in degrees C/W. The default setting is "auto".

-airflow *[0:750]* - (Optional) Linear Feet Per Minute (LFM) airflow to be used for modeling. The default setting varies by device family.

-heatsink *arg* - (Optional) The heatsink profile to be used during modeling. Valid values are: none, low, medium, high, custom. The default setting is "medium".

-thetajb *arg* - (Optional) The Theta-JB thermal resistance used for modeling in degrees C/W. The default setting is "auto".

-board *arg* - (Optional) 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) The board temperature in degrees Centigrade to be used for modeling.

-board_layers *arg* - (Optional) 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 "12to15".

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

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
```

The following example sets the supply voltage Vccaux to a value of 1.9 :

```
set_operating_conditions  -voltage {Vccaux 1.89}
```

The following example sets the manufacturing process corner to maximum:

```
set_operating_conditions -process maximum
```

The following example sets the manufacturing process corner to typical and the voltage supply Vccint to 0.98:

```
set_operating_conditions -process maximum -voltage {Vccint 0.98}
```

See Also

- report_operating_conditions
- report_power
- reset_operating_conditions

set_package_pin_val

Set user columns on one or more package pins.

Syntax

```
set_package_pin_val [-quiet]
[-verbose] column value package_pins ...
```

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
column	User column name
value	Value to set
package_pins	Package pin names

Categories

XDC, PinPlanning

Description

Create user-defined package pin attributes and assign values to specific pins on the package.

User-defined pin attributes can be defined in a CSV file and imported into an I/O Pin Planning project using **read_csv**, or can be edited in the project using this command.

Note Use the **set_property** command to set tool-defined properties of a package pin.

Arguments

column - Specify the user-defined column name. The column name is case-sensitive. If the column does not already exist, a new attribute is created for package pins. If the user-defined column name already exists, the specified value is assigned to the specified pins.

Note Column refers to the display of the attribute in the Package Pins view in the tool GUI. The result of the command is an attribute on the specified package pins that can be exported with **write_csv** for instance.

value - Specify the value to assign to the specified column. You can repeat the **set_package_pin_val** command to assign different values to different pins in the same column.

package_pins - Specify the package pins to assign the value to. You can use the **get_package_pins** command to specify the package pins to set.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example creates a new user-defined column in the Package Pins view, and assigns the value **true** to the specified pin:

```
set_package_pin_val -column track1 -value true -package_pins AK27
```

The following example creates a user-defined column called **Test**, then assigns the value RED to all "AK" package pins, then changes the value to GREEN for the three specified pins:

```
set_package_pin_val -column Test -value RED -package_pins [get_package_pins AK*]
set_package_pin_val -column Test -value GREEN -package_pins {AK1 AK2 AK3}
```

See Also

- get_package_pins
- set_property
- write_csv

set_param

Set a parameter value.

Syntax

set_param [**-quiet**] [**-verbose**] *name value*

Returns

Newly set parameter value

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
name	Parameter name
value	Parameter value

Categories

PropertyAndParameter

Description

Sets the value of a user-definable configuration parameter. These parameters configure and control various behaviors of the tool. Refer to **report_param** for a description of 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.

Arguments

name - The name of the parameter to set the value of. You can only set the value of one parameter at a time.

value - The value to set the specified parameter to.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example sets the value of the tcl.statsThreshold parameter to 15:

```
set_param tcl.statsThreshold 15
```

See Also

- get_param
- list_param
- report_param
- reset_param

set_power_opt

Set constraints for power optimization.

Syntax

```
set_power_opt [-include_cells args] [-exclude_cells args]
[-clocks args] [-cell_types args] [-quiet] [-verbose]
```

Returns

Nothing

Usage

Name	Description
[-include_cells]	Include only these instances for clock gating. Default: all
[-exclude_cells]	Exclude these instances from clock gating. Default: none
[-clocks]	Clock gate instances clocked by these clocks only. Default: all clocks
[-cell_types]	Clock gate these types only. One of [all|bram|reg|srl|none]. Default: all. Default: all
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

Power, XDC

Description

Specify cell instances to include in power optimization. The specified cells are optimized using the **power_opt_design** command.

The effect of multiple **set_power_opt** commands is cumulative, so that you can specify a broad class of cell types to optimize, include specific hierarchical cells, and then exclude cells within the included hierarchy to refine the power optimization.

The power optimizations that have been performed can be reported using the **report_power_opt** command.

Arguments

-include_cells *args* - (Optional) Include only these instances for clock gating. Use this option to list specific cells or blocks to be optimized using **power_opt_design**. The default is to include all cells in power optimization.

-exclude_cells *args* - (Optional) Exclude these instances from clock gating. The default is to not exclude cells from power optimization. The **-exclude_cells** option excludes from the currently included cells. By default all cells are included, however, if **-include_cells** has been specified, then **-exclude_cells** applies only to the currently included cells.

-clocks *args* - (Optional) Perform power optimizations on instances clocked by the specified clocks only. The default is to include all clocks in the design.

Note It is possible to use both **-clocks** and **-include_cells** to produce a list of cells that are not clocked by the specified clocks, resulting in no power optimization

-cell_types [all | bram | reg | slr | none] - (Optional) Perform power optimization on the specified cell types only. The default is to perform power optimization on all types of cells. You can use **all** or **none** to reset, or clear, any prior **set_power_opt** commands.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example sets power optimization for BRAM cells only, and then runs power optimization:

```
set_power_opt -cell_types bram
power_opt_design
```

The following example sets power optimization for BRAM and REG type cells, then adds SRLs, and runs power optimization. Then all cells are cleared, and only SRLs are included, and power optimization is run again:

```
set_power_opt -cell_types { bram reg}
set_power_opt -cell_types { srl}
power_opt_design
set_power_opt -cell_types { none}
set_power_opt -cell_types { srl}
power_opt_design
```

The following example sets power optimization for BRAM cells only, excludes the cpuEngine block from optimization, but then includes the cpuEngine/cpu_dbg_dat_i block, then performs power optimization:

```
set_power_opt -cell_types bram
set_power_opt -exclude_cells cpuEngine
set_power_opt -include_cells cpuEngine/cpu_dbg_dat_i
power_opt_design
```

set_property

Set property on object(s).

Syntax

set_property [**-quiet**] [**-verbose**] *name value objects...*

Returns

The value that was set if success, "" if failure

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
name	Name of property to set
value	Value of property to set
objects	Objects to set properties on

Categories

Object, PropertyAndParameter

Description

Assigns the defined property *name* and *value* to the specified *objects*.

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 on an object, use **report_property**, **list_property**, or **list_property_values**.

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.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

name - Specifies the name of the property to be assigned to the object or objects. The *name* argument is case sensitive and should be specified appropriately.

value - Specifies 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.

Important! In some cases the value of a property may include special characters, such as the dash character ('-'), which can cause the tool to interpret the value as a new argument to the command. In this case, you must use the explicit arguments (**-name**, **-value**, **-objects**) instead of the implied positional arguments (name, value, objects) as described here. This is shown in the **Examples** section below

objects - One or more objects to assign the property to.

Examples

Create a new boolean property, TRUTH, for cell objects, and set the property on a cell:

```
create_property -type bool truth cell
set_property truth false [lindex [get_cells] 1]
```

The following example sets the TOP property of the current fileset to define the top module of the project:

```
set_property top fftTop [current_fileset]
set_property top_file {C:/Data/sources/fftTop.v} [current_fileset]
```

Note Defining the top module requires the TOP property to be set to the desired hierarchical block in the source fileset of the current project. In the preceding example TOP is the property name, fftTop is the value, and current_fileset is the object. In addition, the TOP_FILE property should be defined to point to the data source for the top module

This example shows how to set a property value that includes the dash character, '-'. The dash can cause the tool to interpret the value as a new command argument, rather than part of the value being specified, and will cause an error as shown. In this case, you must use the explicit form of the positional arguments in the set_property command:

```
set_property {XELAB.MORE_OPTIONS} {-pulse_e_style ondetect} [get_filesets sim_1]
ERROR: [Common 17-170] Unknown option '-pulse_e_style ondetect', please type 'set_property
set_property -name {XELAB.MORE_OPTIONS} -value {-pulse_e_style ondetect}\
-objects [get_filesets sim_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 ]
```

The following example configures the synth_1 run, setting options for Vivado Synthesis 2012, and then reports all of the properties of the run:

```
set_property args.synth_design.gated_clock_conversion auto [get_runs synth_1]
set_property args.synth_design.fanout_limit 200 [get_runs synth_1]
set_property flow {Vivado Synthesis 2012} [get_runs synth_1]
report_property -all [get_runs synth_1]
```

See Also

- current_fileset
- 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**] [**-verbose**] *value*

Returns

String result

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
value	Speed grade used for timing analysis

Categories

Project

Description

Sets 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

value - The speed grade for the target device. Valid values are -1, -2, or -3.

Examples

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] [-quiet]
[-verbose] [objects...]
```

Returns

Nothing

Usage

Name	Description
[**-toggle_rate**]	Toggle rate value: 0% <= Value <= 100%. Use with RTL power estimation. Default: 0.0
[**-type**]	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**]	Static probability value: 0 <= Value <= 1 Default: 0.0
[**-signal_rate**]	Signal rate Default: 0.0
[**-hier**]	Hierarchically sets the switching activity on a hierarchical instance provided via option. This option should be used only with option
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*objects*]	Objects to set switching activity on

Categories

XDC, Power

Description

Sets the signal rate and the switching probability to be used when performing power estimation. These include simple signal rate and simple static probability on nets, ports, and pins; and state dependent static probabilities on cells.

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

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 is used to specify a default activity rate for a broad class of signals when performing power estimation. It 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.

Arguments

-toggle_rate *rate* - (Optional) The toggle rate, which 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 0.

-type *value* - (Optional) 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) The switching probability to be used in analysis. Valid values are 0 < value < 1. The default value is 0.

-signal_rate *value* - (Optional) The signal frequency to be used for analysis. The default value is 0.

Note One or both of **-static_probability** or **-signal_rate** must be specified with the **set_switching_activity** command.

-hier - (Optional) Apply the switching activity 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.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

objects - (Optional) A list of port, pin, net, and clock objects to which the switching activity configuration should be applied.

Examples

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_default_switching_activity
- report_power
- report_switching_activity
- reset_default_switching_activity
- reset_switching_activity

show_objects

Show objects in Find Results view.

Syntax

show_objects [**-new**] [**-name** *arg*] [**-quiet**] [**-verbose**] *objects*

Returns

Nothing

Usage

Name	Description
[**-new**]	To add a new tab to Find Results instead of replacing existing one.
[**-name**]	Tab title
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
objects	Objects to show Find Results view

Categories

GUIControl

show_schematic

Show netlist items in schematic view.

Syntax

```
show_schematic [-add] [-remove] [-regenerate] [-pin_pairs]
[-name arg] [-quiet] [-verbose] objects
```

Returns

Nothing

Usage

Name	Description
[-add]	Add to existing schematic view
[-remove]	Remove from existing schematic view
[-regenerate]	Regenerate layout of schematic view
[-pin_pairs]	objects are treated as pair of connected pins. This can be useful to display paths
[-name]	Schematic window title
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
objects	Netlist items to show in schematic view

Categories

GUIControl

Description

Create a schematic view containing the specified design objects when the tool is invoked in GUI mode.

The scope of the schematic that is displayed depends on the objects specified. A schematic created with cells, shows the specified cells and any connections between the cells. A schematic created with pins, shows the pin objects, or shows them connected as appropriate if **-pin_pairs** is specified. A schematic created with nets shows the nets, as well as the cells and ports connected to the nets.

To display a schematic with multiple levels of hierarchy, use the **current_instance** command to set the top-level of the hierarchy, or the level of interest, and use the **-hierarchical** option when specifying design objects with a **get_*** command.

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

Arguments

-add - (Optional) Add the specified objects to the schematic window.

-remove - (Optional) Remove the specified objects from the schematic window.

-regenerate - (Optional) Regenerate the schematic window.

-pin_pairs - (Optional) When specified with a pair of connected pin objects, the schematic shows the pins and the wire between the pins. When the **-pin_pairs** option is not specified, or is specified with disconnected pins, the wire is not shown.

-name *arg* - (Optional) Defines a name for the schematic window opened in the GUI. Use this name to add to, remove from, or regenerate the schematic.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

objects - The netlist objects to display in the schematic window.

Examples

The following example creates a schematic for the top-level of the design, displaying the nets as well as the ports and cells they connect to:

```
show_schematic [get_nets]
```

The following example sets the level of hierarchy of interest, and creates a hierarchical schematic from the current level down:

```
current_instance A
show_schematic [get_nets -hier]
```

The following example creates a schematic window showing the specified pins, and the wire connection between them:

```
show_schematic -pin_pairs [get_pins {data0_i/O data_reg/D}]
```

See Also

- current_instance
- get_cells
- get_nets
- get_pins
- get_ports

split_diff_pair_ports

Remove differential pair relationship between 2 ports.

Syntax

split_diff_pair_ports [**-quiet**] [**-verbose**] *ports*...

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
ports	Ports to split

Categories

PinPlanning

Description

Splits an existing differential pair into two single-ended ports.

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

Arguments

ports - The names of two ports of a differential pair to split into single-ended ports.

Examples

The following example splits the specified diff pair ports to form two single ended ports:

```
split_diff_pair_ports PORT_N PORT_P
```

See Also

- make_diff_pair_ports
- create_port
- create_interface

start_gui

Start GUI.

Syntax

start_gui [**-verbose**]

Returns

Nothing

Usage

Name	Description
[**-verbose**]	Suspend message limits during command execution

Categories

GUIControl

Description

Launches the GUI when the tool is running in the PlanAhead tool Tcl shell. The GUI is invoked with the current project, design, and run information.

Arguments

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example is executed from the command prompt when the tool is running in Tcl 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**] [**-verbose**]

Returns

Int

Usage

Name	Description
[**-try**]	Don't start a group if one has already been started
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

GUIControl

Description

Starts a sequence of commands that can be undone or redone as a series. Use **endgroup** to end the sequence of commands.

Note You can have multiple command groups to **undo** or **redo**, but you cannot nest command groups. You must use **endgroup** to end a command sequence before 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

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
- undo

stop_gui

Close GUI.

Syntax

stop_gui [**-verbose**]

Returns

Nothing

Usage

Name	Description
[**-verbose**]	Suspend message limits during command execution

Categories

GUIControl

Description

Stops GUI mode and places the tool into Tcl mode, running in the PlanAhead tool Tcl shell.

In Tcl mode, all commands must be entered as Tcl commands or through Tcl scripts.

Arguments

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example stops and closes the GUI and places the tool into Tcl mode:

```
stop_gui
```

See Also

start_gui

swap_locs

Swap two locations.

Syntax

swap_locs [**-quiet**] [**-verbose**] *aloc bloc*

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
aloc	First location (port/cell/site - should be of same type as 'bloc')
bloc	Second location (port/cell/site - should be of same type as 'aloc')

Categories

Floorplan

Description

Swaps 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

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.

Examples

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**] [**-verbose**]

Returns

With -list, the list of undoable tasks

Usage

Name	Description
[**-list**]	Show a list of undoable tasks
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

GUIControl

Description

Undo a prior command. 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 - (Optional) Reports the list of commands that can be undone. As you execute the undo command, the tool will step backward through this list of commands.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example returns a list of commands that you can undo:

```
undo -list
```

See Also

- redo
- startgroup
- endgroup

unhighlight_objects

Unhighlight objects that are currently highlighted.

Syntax

```
unhighlight_objects [-color_index arg] [-rgb args] [-color arg]
[-quiet] [-verbose] [objects]
```

Returns

Nothing

Usage

Name	Description
[**-color_index**]	Color index
[**-rgb**]	RGB color index list
[**-color**]	Valid values are red green blue magenta yellow cyan and orange
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*objects*]	Objects to unhighlight

Categories

GUIControl

Description

This command is for use in 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. These options can be used to unhighlight all objects currently highlighted in the specified color. See the example below.

Arguments

-color_index arg - (Optional) 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 - (Optional) 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 - (Optional) Specify the color to unhighlight. Supported highlight colors are red, green, blue, magenta, yellow, cyan, and orange.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

objects - (Optional) 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] [-verbose]
[objects]
```

Returns

Nothing

Usage

Name	Description
[**-rgb**]	RGB color index list
[**-color**]	Valid values are red green blue magenta yellow cyan and orange
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*objects*]	Objects to unmark

Categories

GUIControl

Description

Unmarks the specified object or objects that were previously marked by the **mark_objects** command. This command is for use in GUI mode.

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* - 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* - The color to unmark. Supported highlight colors are red, green, blue, magenta, yellow, cyan, and orange.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

objects - 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

unplace_cell

Unplace one or more instances.

Syntax

unplace_cell [**-quiet**] [**-verbose**] *cell_list* ...

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
cell_list	a list of cells to be unplaced

Categories

Floorplan

Description

Unplace the specified cells from their current placement site.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

cell_list - Specifies a list of one or more cells to be unplaced from the device.

Examples

The following example unplaces the specified cell:

```
unplace_cell {fftEngine/fftInst/ingressLoop[6].ingressFifo/buffer_fifo/i_4773_12897}
```

The following example unplaces multiple cells:

```
unplace_cell {div_cntr_reg_inferredi_4810_15889  div_cntr_reg[0] div_cntr_reg[1]}
```

See Also

place_cell

unselect_objects

Unselect items that are currently selected.

Syntax

unselect_objects [**-quiet**] [**-verbose**] [*objects*]

Returns

Nothing

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
[*objects*]	Objects to unselect

Categories

GUIControl

Description

Unselects the specified object or objects that were previously selected by the **select_objects** command.

This 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 - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

objects - 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_compile_order

Updates a fileset compile order and possibly top based on a design graph.

Syntax

update_compile_order [**-fileset** *arg*] [**-quiet**] [**-verbose**]

Returns

Nothing

Usage

Name	Description
[**-fileset**]	Fileset to update based on a design graph
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Project

Description

Update the compile order of the design sources in the current project, or in the specified fileset.

Arguments

-fileset *arg* - Update the compile order of the specified fileset.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example updates the compile order of the source files in the simulation fileset:

```
update_compile_order -fileset sim_1
```

See Also

- add_files
- import_files

update_files

Update files in the project with same named files from the files or directories specified.

Syntax

```
update_files [-from_files args] [-norecurse] [-to_files args]
[-filesets args] [-force] [-report_only] [-quiet] [-verbose]
```

Returns

List of the files updated

Usage

Name	Description
[**-from_files**]	New files and directories to use for updating
[**-norecurse**]	Recursively search in specified directories
[**-to_files**]	Existing project files and directories to limit updates to
[**-filesets**]	Fileset name
[**-force**]	Overwrite imported files in the project, even if read-only, if possible
[**-report_only**]	Do no actual file updates, but report on updates that otherwise would have been made
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Project

update_ip_catalog

Update the IP Catalog. Before executing this command optionally use the following to set repository paths:'set_property ip_repo_paths [current_fileset]'.

Syntax

```
update_ip_catalog [-rebuild] [-add_ip arg] [-delete_ip arg]
[-repo_path arg] [-quiet] [-verbose]
```

Returns

True for success

Usage

Name	Description
[-rebuild]	Trigger a rebuild of the specified repository's index file or rebuild all repositories if none specified
[-add_ip]	Add the specified IP into the specified repository Values: Either a path to the IP's component.xml or to a zip file containing the IP
[-delete_ip]	Remove the specified IP from the specified repository Values: Either a path to the IP's component.xml or its VLNV
[-repo_path]	Used in conjunction with rebuild, add_ip or delete_ip to specify the path of the repository on which to operate
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution

Categories

IPFlow

Description

Update the IP Catalog associated with the current design.

The default IP catalog is found in the installation hierarchy of the tool. However, you can also add new IP repository paths to the IP catalog using the **set_property** command to define the IP_REPO_PATHS property of the current Source fileset.

The Xilinx® IP catalog, or repository, is located in the installation hierarchy of the software release being used. You can also add custom IP to the repository by using the **set_property** command to set the IP_REPO_PATHS property on the source fileset to point to the locations of custom IP, as shown in the example below.

Arguments

-rebuild - (Optional) Rebuild the complete IP Catalog index, or just rebuild the index for the IP repository specified by the **-repo_path**.

-add_ip *arg* - Add an individual IP core to the specified IP repository. This argument requires the **-repo_path** argument to also be specified. The IP is specified as a path to the `component.xml` of the IP, or the path to a zip file containing the IP.

-delete_ip *arg* - Remove an IP core from the specified IP repository. This argument requires the **-repo_path** argument to also be specified. The IP is specified as a path to the `component.xml` of the IP, or as the VLNV property of the IP. The VLNV property refers to the Vendor:Library:Name:Version string which identifies the IP in the catalog.

-repo_path *arg* - Specify the directory name of an IP repository to add to or delete from, or to rebuild the index for.

Note The IP repository must have been previously added to the current Source fileset using the **set_property** command. See the example below

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example sets the IP_REPO_PATHS property of the current Source fileset, to add an IP repository, then rebuilds the IP catalog index for the whole IP catalog:

```
set_property ip_repo_paths C:/Data/IP_LIB [current_fileset]
update_ip_catalog -rebuild
```

See Also

- create_ip
- import_ip
- generate_target

update_timing

Update timing.

Syntax

update_timing [**-full**] [**-quiet**] [**-verbose**]

Returns

Nothing

Usage

Name	Description
[**-full**]	Perform a full timing update instead of an incremental one
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Timing

upgrade_ip

Upgrade a configurable IP to a later version.

Syntax

```
upgrade_ip [-srcset arg] [-latest arg] [-vlnv arg] [-quiet]
[-verbose] [objects ...]
```

Returns

List of files that were upgraded

Usage

Name	Description
[-srcset]	Source set name
[-latest]	Upgrade the IP to the latest version
[-vlnv]	VLNV string for the Catalog IP to which the IP will be upgraded
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[objects]	IP to be upgraded

Categories

IPFlow

Description

This command upgrades the specified IP cores from an older version to the latest version in the IP catalog.

You can only upgrade legacy IP that explicitly supports upgrading. The UPGRADE_VERSIONS property on the ipdef object indicates if there are upgrade version for an IP core.

Note Not all legacy IP support upgrading, and native IP cannot be upgraded

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

objects - (Optional) Specifies which legacy IP cores should be upgraded.

Examples

The following example upgrades all IP cores in the current project to the latest version:

```
upgrade_ip
```

See Also

- create_ip
- import_ip

verify_config

Analyze implemented runs to ensure they follow rules required for partial reconfiguration.

Syntax

verify_config [**-file** *arg*] [**-quiet**] [**-verbose**] *runs...*

Returns

Pr report

Usage

Name	Description
[**-file**]	Output report file name
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
runs	List of implemented Runs to be verified

Categories

PartialReconfiguration

Description

Analyzes the implemented configurations of a design to ensure they follow rules required for partial reconfiguration.

Arguments

-file *arg* - (Optional) Output the results of this command to the specified log file.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

runs - The list of two or more implemented configuration runs to verify. These specify the runs used to implement the different configurations of the design.

Examples

The following example checks two runs, **config_2** and **config_3**, for any errors and writes the log into the specified log file:

```
verify_config -runs {config_2 config_3} -file pr_verify.log
```

See Also

create_reconfig_module

version

Returns the build for PlanAhead and the build date.

Syntax

version [**-short**] [**-quiet**] [**-verbose**]

Returns

PlanAhead version

Usage

Name	Description
[**-short**]	Return only the numeric version number
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution

Categories

Report

Description

Returns the version number of the Xilinx® tool. This includes the software version number, build number and date, and copyright information.

Arguments

-short - (Optional) Returns the version number of the software only.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

The following example returns only the version number for the software:

```
version -short
```

wait_on_run

Block execution of further Tcl commands until the specified run completes.

Syntax

wait_on_run [**-timeout** *arg*] [**-quiet**] [**-verbose**] *run*

Returns

Nothing

Usage

Name	Description
[**-timeout**]	Maximum time to wait for the run to complete (in minutes) Default: -1
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
run	Run to wait on

Categories

Project

Description

Blocks the execution of Tcl commands until the specified run completes, or until the specified amount of time has elapsed.

The **wait_on_run** command can be used for runs that have been launched. If the specified run has not been launched when the **wait_on_run** command is used, you will get an error. Runs that have already completed do not return an error.

Note This command is used for running the tool in batch mode or from Tcl scripts. It is ignored when running interactively from the GUI.

Arguments

-timeout *arg* - 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 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 tool will wait for the run to complete.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

run - The name of the run to wait on.

Examples

The following example launches the impl_1 run, and then waits 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_chipscope_cdc

Export nets that are connected to debug ports.

Syntax

write_chipscope_cdc [**-quiet**] [**-verbose**] *file*

Returns

Name of the output file

Usage

Name	Description
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
file	ChipScope CDC file name

Categories

FileIO, ChipScope

Description

Writes 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 buy using the **read_chipscope_cdc** command.

Arguments

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

file - The CDC file name to be written.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Examples

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 [**-force**] [**-quiet**] [**-verbose**] *file*

Returns

Name of the output file

Usage

Name	Description
[**-force**]	Overwrite existing file
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
file	Pin Planning CSV file

Categories

FileIO

Description

Writes 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 - Overwrite the CSV file if it already exists.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

file - The filename of the CSV file to write.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Examples

The following example exports a CSV file from the current project:

```
write_csv C:/Data/pinList.csv
```

See Also

read_csv

write_debug_probes

Write debug probes to a file.

Syntax

```
write_debug_probes [-force] [-quiet] [-verbose] file
```

Returns

Name of the output file

Usage

Name	Description
[**-force**]	Overwrite existing file
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
file	Debug probes file name (default extension is .ltx)

Categories

FileIO, ChipScope

Description

Writes an ILA probes file that contains information about the nets that you probed in the current design using the ILA v1.0 core. The debug probes data file typically has a `.ltx` file extension.

ILA cores are added to the design using the **create_debug_core** command, and connected to nets in your design using the **connect_debug_core** command.

The specific information and use of the debug probes file is described in the *ChipScope Pro Software and Cores User Guide (UG029)*.

Arguments

-force - (Optional) Overwrite the specified file if it already exists.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

file - (Required) The file name of the debug probes file to write.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Examples

The following example write a debug probe file from the current design:

```
write_debug_probes C:/Data/designProbes.ltx
```

See Also

- connect_debug_port
- create_debug_core
- create_debug_port
- implement_debug_core

write_edif

Export the current netlist as an EDIF file.

Syntax

```
write_edif [-pblocks args] [-cell arg] [-force]
[-security_mode arg] [-quiet] [-verbose] file
```

Returns

The name of the output file or directory

Usage

Name	Description
[**-pblocks**]	Export netlist for these pblocks (not valid with -cell)
[**-cell**]	Export netlist for this cell (not valid with -pblocks)
[**-force**]	Overwrite existing file
[**-security_mode**]	If set to 'all', and some of design needs encryption then whole of design will be written to a single encrypted file Default: multifile
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
file	Output file (directory with -pblocks or -cell)

Categories

FileIO

Description

Writes the current netlist as an EDIF file, or outputs the contents of specific Pblocks or hierarchical cells as EDIF netlist files.

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 tool will return an error.

Arguments

-pblocks *arg* - (Optional) Instructs the 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* - (Optional) Instructs the 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 - (Optional) Overwrite the EDIF file if it already exists.

-security_mode [multifile | all] - (Optional) Write a multiple EDIF files when encyrpted IP is found in the design, or write a single file.

- **multifile** - This is the default setting. By default the command writes out the full design netlist to the specified file. However, if the design contains secured IP, it creates an encrypted file containing the contents of the secured module. This will result in the output of multiple EDIF files, containing secured and unsecured elements of the design.

- **all** - Write both encrypted and unencrypted cells to a single specified file.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

file - The filename of the EDIF file to write. The default file extension for an EDIF netlist is .edn. If the **-pblocks** or **-cell** options are used, the name specified here refers to a directory rather than a single file.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

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/
```

See Also

read_edif

write_ibis

Write IBIS models for current floorplan.

Syntax

```
write_ibis [-force] [-allmodels] [-nopin] [-truncate arg]
[-ibs arg] [-pkg arg] [-quiet] [-verbose] file
```

Returns

Name of the output file

Usage

Name	Description
[**-force**]	Overwrite existing .ibs file
[**-allmodels**]	Include all available buffer models for this architecture. By default, only buffer models used by the floorplan are included.

Categories

FileIO

Description

Writes the IBIS models for the target device in the current design. The netlist and implementation details from the design are combined 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 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 file will include per-pin modeling of the package as RLC matrices in the [Define Package Model] section if this data is available.

-truncate *arg* - The maximum length for a signal name in the output file. Names longer than this will be truncated. Valid values are 20, 40, or 0 (unlimited). By default the signal names are truncated 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 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 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 - The filename of the IBIS file to write.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

Examples

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_timing

Export a set of timing results to file.

Syntax

write_timing [**-force**] [**-quiet**] [**-verbose**] *name file*

Returns

Nothing

Usage

Name	Description
[**-force**]	Overwrite existing file
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
name	Name for the set of results
file	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 - Overwrite the timing file if it already exists.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

name - 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 - The filename of the timing file to write.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Examples

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**] [**-verbose**] *file*

Returns

Name of the output file or directory

Usage

Name	Description
[**-no_fixed_only**]	Export fixed and non-fixed placement (by default only fixed placement will be exported)
[**-constraints**]	Include constraints that are flagged invalid Values: valid, invalid, all Default: valid
[**-pblocks**]	Export placement for these pblocks (not valid with -cell)
[**-cell**]	Export placement for this cell (not valid with -pblocks)
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
file	Output file (directory with -pblocks, -cell)

Categories

FileIO

Description

Writes physical constraints to a user constraint file (UCF). Constraints can be written from the top-level, which is the default, from specific Pblocks, or from a specific hierarchical cell.

This command will write 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 - Write 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* - Write 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* - One or more Pblocks from which to write the constraints.

-cell *arg* - 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 this option.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

file - The filename of the UCF file to write.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Note When you use **-pblock** or **-cell**, this argument specifies a directory name where the UCF files for each Pblock or cell will be written. Each UCF file will be named after a Pblock or cell. If the specified directory does not exist you will get an error.

Examples

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

write_verilog

Export the current netlist in Verilog format.

Syntax

```
write_verilog [-cell arg] [-mode arg] [-lib]
[-port_diff_buffers] [-write_all_overrides] [-rename_top arg]
[-sdf_anno arg] [-sdf_file arg] [-force] [-include_xilinx_libs]
[-quiet] [-verbose] file
```

Returns

The name of the output file or directory

Usage

Name	Description
[**-cell**]	Root of the design to write. eg. des.subblk.cpu Default: whole design
[**-mode**]	Values: design, port, sta, funcsim, timesim Default: design
[**-lib**]	Write each library into a separate file
[**-port_diff_buffers**]	Output differential buffers when writing in -port mode
[**-write_all_overrides**]	Write parameter overrides on Xilinx primitives even if the override value is the same as the default value
[**-rename_top**]	Replace top module name with custom name e.g. netlist Default: new top module name
[**-sdf_anno**]	Specify if sdf_annotate system task statement is generated
[**-sdf_file**]	Full path to sdf file location Default: .sdf
[**-force**]	Overwrite existing file
[**-include_xilinx_libs**]	Include simulation models directly in netlist instead of linking to library
[**-quiet**]	Ignore command errors
[**-verbose**]	Suspend message limits during command execution
file	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* - The mode to use when writing the Verilog file. By default, the Verilog netlist is written for the whole design. Valid 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).

• funcsim - Output a Verilog netlist to be used for functional simulation. The output netlist is not suitable for synthesis.

• timesim - Output a Verilog netlist to be used for timing simulation. The output netlist is not suitable for synthesis.

-lib - Create a separate Verilog file 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

-port_diff_buffers - Add the differential pair buffers and internal wires associated with those buffers into the output ports list. This argument is only valid when **-mode port** is specified.

-write_all_overrides [true | false] - Write parameter overrides, in the design 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* - Rename the top module in the output as specified. This option only works with **-mode funcsim** or **-mode timesim** to allow the Verilog netlist to plug into top-level simulation test benches.

-sdf_anno [true | false] - Add the **$sdf_annotate** statement to the Verilog netlist. Valid values are true (or 1) and false (or 0). This option only works with **-mode timesim**, and is set to false by default.

-sdf_file *arg* - The path and filename of the SDF file to use when writing the `$sdf_annotate` statement into the output 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 - Overwrite the Verilog files if they already exists.

-include_xilinx_libs - This option writes the simulation models directly in the output netlist file rather than pointing to the libraries by reference.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

file - The filename of the Verilog file to write. 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 file will be written into the current working directory, or the directory from which the 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 timesim** 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 timesim -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_vhdl

write_vhdl

Export the current netlist in VHDL format.

Syntax

```
write_vhdl [-cell arg] [-mode arg] [-lib] [-port_diff_buffers]
[-write_all_overrides] [-rename_top arg] [-arch_only] [-force]
[-include_xilinx_libs] [-quiet] [-verbose] file
```

Returns

The name of the output file or directory

Usage

Name	Description
[-cell]	Root of the design to write. eg. des.subblk.cpu Default: whole design
[-mode]	Output mode. Valid values: funcsim, port Default: funcsim
[-lib]	Write each library into a separate file
[-port_diff_buffers]	Output differential buffers when writing in -port mode
[-write_all_overrides]	Write parameter overrides on Xilinx primitives even if the same as the default value
[-rename_top]	Replace top module name with custom name e.g. netlist Default: new top module name
[-arch_only]	Write only the architecture, not the entity declaration for the top cell
[-force]	Overwrite existing file
[-include_xilinx_libs]	Include simulation models directly in netlist instead of linking to library
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
file	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* - The mode to use when writing the VHDL file. By default, the simulation netlist is written for the whole design. Valid mode values are:

- funcsim - Output the VHDL netlist to be used as a functional 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 module.

-lib - Create a separate VHDL file 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

-port_diff_buffers - Add the differential pair buffers and internal wires associated with those buffers into the output ports list. This argument is only valid when **-mode port** is specified.

-write_all_overrides [true | false] - Write parameter overrides in the design 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* - Rename the top module in the output as specified. This option only works with **-mode funcsim** to allow the VHDL netlist to plug into top-level simulation test benches.

-arch_only - Suppress the entity definition of the top module, and outputs the architecture only. This simplifies the use of the output VHDL netlist with a separate test bench.

-include_xilinx_libs - Write the simulation models directly in the output netlist file rather than pointing to the libraries by reference.

-force - Overwrite the VHDL files if they already exists.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

file - The filename of the VHDL file to write. 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 file will be written into the current working directory, or the directory from which the 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

Writes constraints to a Xilinx Design Constraints (XDC) file. The default file extension for a XDC file is .xdc.

Syntax

```
write_xdc [-no_fixed_only] [-constraints arg] [-cell arg]
[-sdc] [-no_tool_comments] [-force] [-exclude_timing] [-quiet]
[-verbose] [file]
```

Returns

Nothing

Usage

Name	Description
[-no_fixed_only]	Export fixed and non-fixed placement (by default only fixed placement will be exported)
[-constraints]	Include constraints that are flagged invalid Values: valid, invalid, all Default: valid
[-cell]	Export placement for this cell.
[-sdc]	Export all timing constraints.
[-no_tool_comments]	Don't write verbose tool generated comments to the xdc when translating from ucf.
[-force]	Overwrite existing file.
[-exclude_timing]	Don't export timing constraints.
[-quiet]	Ignore command errors
[-verbose]	Suspend message limits during command execution
[*file*]	Output constraints to the specified XDC file.

Categories

FileIO

Description

Writes constraints to a Xilinx Design Constraints file (XDC). The XDC can be exported from the top-level, which is the default, 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. Valid values are VALID, INVALID, or ALL.

-cell *arg* - 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 this option.

-sdc - Export only the timing constraints in an SDC format from the current design. Does not export any other defined constraints.

-no_tool_comments - Do not generate tool comments when writing the XDC file. Without this argument, comments and warnings related to the creation of the XDC file will be added.

-force - Overwrite the XDC file if it already exists.

-quiet - (Optional) Execute the command quietly, returning no messages from the command. The command also returns TCL_OK regardless of any errors encountered during execution.

Note Any errors encountered on the command-line, while launching the command, will be returned. Only errors occurring inside the command will be trapped.

-verbose - (Optional) Temporarily override any message limits and return all messages from this command.

Note Message limits can be defined with the **set_msg_limit** command.

file - The filename of the XDC file to write.

Note If the path is not specified as part of the file name, the file will be written into the current working directory, or the directory from which the tool was launched.

Examples

The following example writes the valid and invalid constraints, including both fixed and unfixed cells, to the specified file:

```
write_xdc -no_fixed_only -constraints all C:/Data/design.xdc
```

Additional Resources

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see the Xilinx Support website at:

http://www.xilinx.com/support

For a glossary of technical terms used in Xilinx documentation, see:

http://www.xilinx.com/company/terms.htm

Solution Centers

See the Xilinx Solution Centers for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

References

Vivado Design Suite 2012.2 Documentation:

www.xilinx.com/support/documentation/dt_vivado2013-1.htm

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/tcl8.5/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