SIEMENS EDA

PowerPro® Reference Manual

Software Version 2023.1
February 2023

SIEMENS

Table of Contents

Chapter 1. Introduction

1.1 Organization

PowerPro Reference Manual is organized into the following chapters:

- Command Reference describes the commands that are used within the Tcl command shell of PowerPro.
- Global Variable Reference describes the global variables that are used within Tcl command shell of PowerPro.
- Command Line Interface describes the different options that can be used in the command line with PowerPro.
- Design Representation describes how design descriptions are represented in PowerPro.
- Pattern Matching Syntax describes the pattern matching syntax used by some of the PowerPro commands.

1.2 Syntax Conventions

The following syntax conventions are used in this manual:

literal	Literal command text
<value>	Value supplied by user
[a]	Optional value
{a}	Required value
a | b	Choice of values
a...	Repeatable value
\	Line continuation

1.3 Feedback Request

You can provide feedback based on your overall experience with PowerPro.

You can also send your feedback to https://account.sw.siemens.com/en-US/help/support.

1.4 Siemens Support

Global Customer Support and Success A support contract with Mentor, a Siemens Business, is a valuable investment in your organization's success. With a support contract, you have 24/7 access to the comprehensive and personalized Support Center portal.

Support Center features an extensive knowledge base to quickly troubleshoot issues by product and version. You can also download the latest releases, access the most up-to-date documentation, and submit a support case through a streamlined process.

https://support.sw.siemens.com

If your site is under a current support contract, but you do not have a Support Center login, register here:

https://support.sw.siemens.com/register

Chapter 2. Command Reference

all_clocks

all_clocks — Returns all clocks in the design. Note that this command can only be called after `prototype_design`.

Synopsis

```
all_clocks
```

This command has no arguments.

Related Commands

all_inputs, all_outputs, all_registers, create_clock, create_generate_clock, get_clocks, report_clocks

Usage

The **all_clocks** command returns a list of clocks in the design defined using the `create_clock` or `create_generated_clock` command.

Examples

Example 1: Returns a list of clocks in the design.

```
all_clocks
```

Example 2: Prints switching activity information for all clock ports in the design.

```
foreach port [all_clocks] {
    set td_val [get_sa  -port $port  -td]
    set prob_val [get_sa  -port $port  -prob]
    puts "$port : td=$td_val  prob=$prob_val"
}
```

all_inputs

all_inputs — Returns a list of all input and inout ports in the design. Note that this command can only be called after `build_design`.

Synopsis

```
all_inputs
```

This command has no arguments.

Related Commands

all_outputs, all_registers, find

Usage

The **all_inputs** command returns a list of input and inout ports in the design. This command is usually used as part of another command that applies constraints on input ports (See Example 2).

Examples

Example 1: Returns a list of all input and inout ports in the design.

```
all_inputs
```

Example 2: Prints switching activity information for all input and inout ports in the design.

```
foreach port [all_inputs] {
    set td_val [get_sa  -port $port  -td]
    set prob_val [get_sa  -port $port  -prob]
    puts "$port : td=$td_val  prob=$prob_val"
}
```

all_outputs

all_outputs — Returns a list of all output and inout ports in the design. Note that this command can only be called after `build_design`.

Synopsis

```
all_outputs
```

This command has no arguments.

Related Commands

all_inputs, find

Usage

The **all_outputs** command returns a list of output and inout ports in the design. This command is usually used as a part of another command that applies constraints on output ports (See Example 2).

Examples

Example 1: Returns a list of all output and inout ports in the design.

```
all_outputs
```

Example 2: Prints switching activity information for all output and inout ports in the design.

```
foreach port [all_outputs] {
    set td_val [get_sa  -port $port  -td]
    set prob_val [get_sa  -port $port  -prob]
    puts "$port : td=$td_val  prob=$prob_val"
}
```

---

all_registers

all_registers — Returns a list of all flops and latches in the design. This command can also be used to return their data pins, clock pins and output pins. Note that this command can only be called after `build_design`.

Synopsis

```
all_registers [ -cells] [ -no_hierarchy] [ -level_sensitive]
    [ -edge_triggered] [ -data_pins] [ -output_pins] [ -clock_pins]
```

-cells

Filters the list to include flops, latches or pins matching the options specified.

Note: If this options is specified along with any of the *<*_pins>* options, the `all_registers` command returns pins of the selected cells.

-no_hierarchy

Restricts the scope of the search to the current hierarchy. Sub -level hierarchies are ignored. If not specified, the entire hierarchical design is searched.

-level_sensitive

Filters the list to include level-sensitive cells or cell pins.

-edge_triggered

Filters the list to include edge-triggered cells or cell pins.

-data_pins

Filters the list to include data pins for the selected cells.

-output_pins

Filters the list to include output pins for the selected cells.

-clock_pins

Filters the list to include clock pins for the selected cells.

Related Commands

all_clocks, all_inputs, all_outputs, find

Usage

The **all_registers** command returns a list of flops or latches in the design. When specified with any of the `*_pins` options, this command returns a list of data pins, clock pins and output pins in the design or a specific hierarchy.

Examples

Example 1: Returns a list of flops and latches in the design.

```
all_registers
```

Example 2: Returns a list of clock pins for all latches in the design.

```
all_registers  -level_sensitive  -clock_pins
```

Example 3: Prints switching activity information for all outputs of flops and latches in the design.

```
foreach port [all_registers  -output_pins] {
    set td_val [get_sa  -port $port  -td]
    set prob_val [get_sa  -port $port  -prob]
    puts "$port : td=$td_val  prob=$prob_val"
}
```

analyze_mem_sim_traces

Deprecated w.e.f. PowerPro 10.4.

analyze_mem_sim_traces — Performs simulation-based analysis on the memories to generate information about possible memory power savings. Note that this command can only be called after the `prototype_design` command and before the `insert_observability_logic` and `insert_stability_logic` commands.

Synopsis

```
analyze_mem_sim_traces { -observability | -sleep | -skid | -all }
```

-observability

Performs a combination of observability-based sequential analysis and simulation-based analysis to find redundant reads being performed on memories in the design.

Note: The `insert_mem_observability_gating` command must have been successfully executed prior to using this option.

-sleep

Performs simulation-based light sleep analysis to find opportunities to put memories into low power light sleep mode.

-skid

Performs simulation-based skid analysis on the memory operations to identify redundant reads and writes performed on memories in the design.

Redundant writes are reported for a memory address only when a write operation occurs before the next read operation.

Note: Redundant writes will only be reported if the above mentioned condition holds true for all memory addresses.

-all

Performs observability and sleep based analysis. Note that a skid analysis is not performed. To perform a skid analysis, use the `-skid` option.

-stability

This option has been deprecated w.e.f PowerPro 7.1. Stability based sequential analysis to identify redundant reads on memories is now performed by default during Cycle Based Analysis.

Related Commands

set_powerpro_mode, read_fsdb, report_memory_redundant_access, insert_observability_logic, insert_stability_logic

Usage

The `analyze_mem_sim_traces` command performs a combination of sequential analysis and simulation-based analysis to find memories in the design where power savings are possible. The redundant memory accesses found by analysis is an upper-bound on the possible savings that could be determined by PowerPro. Note that PowerPro might not be able to disable all redundant memory accesses. Information about leftover redundant memory accesses can prove useful when performing changes to increase power savings.

Note: The `analyze_mem_sim_traces` command must be issued before the `report_memory_redundant_access` command to report information about redundant memory accesses.

Examples

Example 1: Performs observability and sleep based analysis.

```
insert_mem_observability_gating
analyze_mem_sim_traces -all
```

Note: The `insert_mem_observability_gating` command is required to perform the observability analysis.

Example 2: Finds all unobservable memory reads.

```
insert_mem_observability_gating
analyze_mem_sim_traces -observability
```

Example 3: Finds unobservable opportunities, as well as opportunities to put memories into low power light sleep mode.

```
insert_mem_observability_gating
analyze_mem_sim_traces -observability -sleep
```

Example 4: Finds all redundant memory reads and writes by performing a skid analysis.

```
analyze_mem_sim_traces -skid
```

analyze_redundant_reset

analyze_redundant_reset — Performs an analysis of the design to identify flops that have redundant resets. This analysis is independent of the switching-activity information. Note that this command can be called only after the `prototype_design` command.

Synopsis

```
analyze_redundant_reset [ -datapath ] [ -maxtime <time_limit> ]
```

-datapath

Reports redundant reset opportunities for only those flops that exist in the data path in the design.

-maxtime <*time_limit*>

Specifies the maximum time in seconds for which the command must be executed to report redundant reset opportunities for flops.

Related Commands

report_redundant_reset_flops, insert_observability_logic

Usage

Performs an analysis of the design to identify flops that have redundant resets. This analysis is independent of the switching-activity information. In other words, changing the fsdb/saif file will not result in any changes in the number of flops that have redundant resets.

Example

Example 1: Performs an analysis of the design file `input.v` to identify flops that have redundant resets.

```
    build_design input.v
    prototype_design
    analyze_redundant_reset
    report_redundant_reset_flops
```

---

bind_to_tech_cell

bind_to_tech_cell — Binds modules to their technology cell definitions if both RTL and technology cell definitions exist. By default, modules are bound to their RTL definition. Note that this command can only be called before `build_design` or `link_design`.

Synopsis

`bind_to_tech_cell` *`<module_list>`*

`<module_list>`

Space separated Tcl list specifying modules to bind to technology cell definitions.

Related Commands

read_library, build_design

Usage

By default, modules are bound to their RTL definitions. To bind modules to their technology cells instead, use the `bind_to_tech_cell` command. This command must be called prior to linking the design using the `build_design` or `link_design` command.

Examples

Example 1: Explicitly binds all instances of the modules `mod1`, `mod2`, and `mod3` to its technology cell definition.

```
bind_to_tech_cell {mod1}
bind_to_tech_cell {mod2}
bind_to_tech_cell {mod3}
```

build_db

build_db — Builds a database representing the design library.

Synopsis

```
build_db
```

Related Commands

build_design, link_design, read_design

Usage

The `build_db` command builds a database from one or all linked and elaborated designs. Note that an error will be issued if the design library has not been linked or has already been built.

Examples

Example 1: Reads the design file `input.v`, links the design and then builds the database.

```
powerpro>read_design input.v
powerpro>link_design
powerpro>build_db
```

build_design

build_design — Reads design files into a design library, links them, and builds a design database.

Synopsis

```
build_design [ -verilog | -vhdl | -sv ] [ -top] [ -cons] <arguments>...
```

-verilog | -vhdl | -sv

Specifies the design language of the files being read in. If omitted, the suffix of the first file read in is used to establish the design language (Verilog if `.v`, VHDL if `.vhd or .vhdl`, and SystemVerilog if `.sv`).

-top

Specifies the top module or entity of the design. A top module is a module that is not instantiated by any other module in the design. The `-top` option must be explicitly specified when multiple top modules exist in the design.

-cons

Adds modules to the constraints design library. These constraints are used by the `set_dont_care`, `set_cgic_cell` and `insert_cg_marker_cell` commands.

-mem

The `-mem` option has been deprecated w.e.f PowerPro 7.1. Memory wrappers with the `*calypto_memory_view*` pragma are now recognized by default. This option no longer needs to be explicitly specified.

Verilog and SystemVerilog Arguments

+define+*<macro>* [=*<macro_body>*]

Defines a macro. For example, to assign the value `1024` to the macro `WIDTH` when the file `input.v` is read in, specify the following command:

```
build_design +define+WIDTH=1024 input.v
```

-f*<list_file>*

Uses a list file to identify the files and options to be read in.

-y*<dir_name>*

Searches for unresolved modules in the directory *<dir_name>*.

-v*<file_name>*

Searches for unresolved modules in the file *<file_name>*.

-defparam *<list_of_parameter_value_pairs>*

Specifies the list of parameter value pairs to override default values for the parameters of the top level module. For example, to override the values of parameters 'p1', 'WIDTH' and ‘`val`’ in the top module of the design, specify:

```
build_design -defparam "{p1 1} {WIDTH $width} {val "\"FIRST\""}"
cpu.v
```

+libext+*<ext>*

Searches for unresolved modules in a library directory using extension *<ext>*.

+librescan

Searches from the beginning of the library list for undefined modules, and forces multiple searches in the same library.

+incdir+*<dir>* ...

Searches + separated directories to resolve `include directives.

<filenames> ...

A whitespace separated list of design files to be read into PowerPro which may include absolute or relative directory paths. By default, any additional file references included within the design files are searched for in the current working directory, and additional include paths may be included using `+incdir+`*<dir>* options.

VHDL Arguments

-87

Specifies that the files should be read in using VHDL 87 standards. Unless specified, the files will be read in using VHDL 93 standards.

-library *<library>*

Specifies that the VHDL units being read in belong to the VHDL logical library *<library>*.

-defparam *<list of generic-value pairs>*

Specifies the list of generic value pairs to override default values for the generics of the top-level entity.

For example, to override the values of generics `p1` and `WIDTH` in the top-entity of the specification design:

```
build_design -spec -defparam "{p1 1} {WIDTH $width}" cpu.vhd
```

Related Commands

build_db, link_design, read_design, read_db

Usage

The `build_design` command is an alias for reading in a design, linking it, and building the design database.

Examples

Example 1: The following example shows how to read in the Verilog design file `input.v` into a design library, link it, and then build a design database.

```
build_design input.v
```

The above mentioned `build_design` command is equivalent to specifying the following commands:

```
read_design input.v
link_design
build_db
```

Example 2: The following example shows how to read in design files listed in the file `input_list.f`.

```
build_design -verilog -f input_list.f
+define+FOR_LOGICBENCH+FOR_SYNTHESIS
```

This command reads in the files listed in `input_list.f` into a design library, links them, and builds a design database. It also defines the verilog macros `FOR_LOGICBENCH` and `FOR_SYNTHESIS`.

Example 3: The following example shows how to read in a design file as well as define the value for a macro that is used by the design file.

```
build_design +define+WIDTH=1024 input.v
```

In this example, the Verilog design file `input.v` is read and the value for the macro `WIDTH` is defined for use with `input.v`.

Example 4: For SystemVerilog, the design file can use a `.v` or `.sv` extension. In case the file uses a `.v` extension, the `-sv` option needs to be explicitly specified to build the design. In this example, `build_design` reads in a SystemVerilog file and builds the database into the PowerPro design library.

```
build_design -sv  design.v
```

config_clock_tree

config_clock_tree — Builds a balanced clock tree for each clock network to compute the power consumed by the nets and cells in the tree.

Synopsis

```
config_clock_tree -buffer <buffer_cell> -fanout<count>
    [ -final_buffer <final_buffer_cell> ]
    [ -root_buffer <root_buffer_cell> ]
    [ -final_buffer_fanout <final_fanout_count> ]
    [ -root_buffer_fanout <root_fanout_count> ]
    [ -inst <hierarchical_instance> ]
```

-buffer *<buffer_cell>*

Specifies the name of the buffer technology cell to be used for clock tree synthesis.

-fanout *<count>*

Specifies the maximum number of fanouts for a buffer in the clock tree.

-final_buffer *<final_buffer_cell>*

Specifies the name of the buffer technology cell that drives the output of the clock tree. If not specified, the technology cell specified by *<buffer_cell>* is used.

-root_buffer *<root_buffer_cell>*

Specifies the name of the buffer technology cell that is placed at the root of the clock tree. If not specified, the technology cell specified by *<buffer_cell>* is used.

-final_buffer_fanout *<final_fanout_count>*

Specifies the maximum number of fanouts for a buffer that drives the output of the clock tree. If not specified, the fanout count specified by *<count>* is used.

-root_buffer_fanout *<root_fanout_count>*

Specifies the maximum number of fanouts for a buffer placed at the root of the clock tree. If not specified, the fanout count specified by *<count>* is used.

-inst *<hierarchical_instance>*

Applies user constraints at the respective hierarchical levels in the design. The user constraints applied to any hierarchical instance are automatically applied to its lower hierarchies successively till a hierarchical instance is reached for which the `config_clock_tree` constraint parameters already exist.

Note: If the option `-inst` is not used, the `config_clock_tree` constraints are applied only to the top hierarchy in the design.

Related Commands

report_clock_tree_power, report_power

Usage

The `config_clock_tree` command instructs PowerPro to build a virtual balanced clock tree for each clock network to compute the power consumed by the nets and cells in the tree.

The root of a clock network is a clock pin, defined by `create_clock` or an internal clock root created by PowerPro, and the leaves are either clock pins of sequential elements (registers/latches/memories), primary output ports or black box input ports.

Integrated clock gating cells are part of a clock network and the tool builds a combined tree including all the clock gating cells and its fanout clock tree. It can handle multiple levels of clock gating cells in the clock network.

Additionally, this command generates information about clock network power that gets displayed in the Power Analysis report. Use the `report_power` and `report_clock_tree_power` commands to view this information.

Example

Example 1: Creates a virtual balanced clock tree using the buffer technology cell `BUF1`. All inserted buffers will drive a maximum of 4 instances.

```
config_clock_tree -buffer BUF1 -fanout 4
```

Note: Instances refer to registers, latches, memories or a primary output ports (including black box input ports) of the clock tree.

Example 2: Creates a virtual balanced clock tree using the buffer technology cell `BUF1` and `BUF2`, where the buffer `BUF2` is only used for final(leaf) buffers. The final buffer will drive a maximum of 10 instances while the remaining buffers will drive a maximum of 4 instances.

```
config_clock_tree -buffer BUF1 -fanout 4 -final_buffer BUF2
        -final_buffer_fanout 10
```

Example 3: Creates a virtual balanced clock tree using the buffer technology cell `BUF1`, `BUF2` and `BUF3`. Here, the buffer `BUF2` is only used for final(leaf) buffers while the buffer `BUF3` is only used for root buffers. The final buffer will drive a maximum of 10 instances, the root buffer will drive a maximum of 2 instances while the remaining buffers will drive a maximum of 4 instances.

```
config_clock_tree -buffer BUF1 -fanout 4 -final_buffer BUF2
        -final_buffer_fanout 10 -root_buffer BUF3 -root_buffer_fanout 2
```

config_design_bin

config_design_bin — Appends or overwrites the existing values passed with the utilities `vlog`, `vopt`, and `vcom` during the generation of the `design.bin` file.

For example, to bypass the suppressible errors, the command can be used to specify values for the given utilities during the generation of the `design.bin` file.

Synopsis

```
config_design_bin
```

```
    [-add_vlog_options <vlog_options> | -set_vlog_options
    <vlog_options>]
    [-add_vopt_options <vopt_options> | -set_vopt_options
    <vopt_options>]
    [-add_vcom_options <vcom_options> | -set_vcom_options
    <vcom_options>]
```

-add_vlog_options <*vlog_options*> | -set_vlog_options <*vlog_options*>

Specifies the arguments that can be used to pass values with the `vlog` utility.

Specify the `-add_vlog_options` argument to append the specified value to the existing value passed with the `vlog` utility.

Specify the `-set_vlog_options` argument to overwrite the existing value passed with the `vlog` utility with the specified value.

Note: The arguments `-add_vlog_options` and `-set_vlog_options` are mutually exclusive.

-add_vopt_options <*vopt_options*> | -set_vopt_options <*vopt_options*>

Specifies the arguments that can be used to pass values with the `vopt` utility.

Specify the `-add_vopt_options` argument to append the specified value to the existing value passed with the `vopt` utility.

Specify the `-set_vopt_options` argument to overwrite the existing value passed with the `vopt` utility with the specified value.

Note: The arguments `-add_vopt_options` and `-set_vopt_options` are mutually exclusive.

-add_vcom_options <*vcom_options*> | -set_vcom_options <*vcom_options*>

Specifies the arguments that can be used to pass values with the `vcom` utility.

Specify the `-add_vcom_options` argument to append the specified value to the existing value passed with the `vcom` utility.

Specify the `-set_vcom_options` argument to overwrite the existing value passed with the `vcom` utility with the specified value.

Note: The arguments `-add_vcom_options` and `-set_vcom_options` are mutually exclusive.

Related Commands

config_guidance_reports

Usage

Appends or overwrites the values passed with the utilities `vlog`, `vopt`, and `vcom`.

Examples

Example 1: Appends the specified value to the existing value passed with the `vlog` utility.

```
config_design_bin -add_vlog_options " -warning 2583,2600,2697,12003 "
```

Example 2: Appends the specified value to the existing value passed with the `vopt` utility.

```
config_design_bin -add_vopt_options " -warning 13259,12345,12348 "
```

Example 3: Appends the specified value to the existing value passed with the `vcom` utility.

```
config_design_bin -add_vcom_options " -warning 13259 "
```

Example 4: Overwrites the existing value passed with the `vlog` utility with the specified value.

```
config_design_bin -set_vlog_options " -mfcu -skipsynthoffregion -
permissive "
```

Example 5: Overwrites the existing value passed with the `vopt` utility with the specified value.

```
config_design_bin -set_vopt_options " -mfcu +no_comp_dir "
```

Example 6: Overwrites the existing value passed with the `vcom` utility with the specified value.

```
config_design_bin -set_vcom_options " -skipsynthoffregion -permissive "
```

config_guidance_reports

config_guidance_reports — Configures the behavior of guidance reports. The command can be run only before the `build_design` command and cannot be run after any optimization command is run.

Synopsis

```
config_guidance_reports
[-report_user_enabled_flops <true | false>]
[-report_blcg <true | false>] [-blcg_flop_ratio_threshold <threshold>]
[-report_sr2cb <true | false>] [-report_access_profile <true | false>]
[-ap_mem_addr_access_threshold <threshold>]
[-compute_access_profile_redund <true | false>]
[-report_bypass_potential <true | false>]
[-compute_bypass_redund <true | false>]
[-report_redundant_mux <true | false>]
[-compute_mux_redund <true | false>]
[-report_high_togg_signals <true | false>]
[-report_clk_toggle_data_stable <true | false>]
[-compute_clk_toggle_data_stable_redund <true | false>]
[-report_redund_mem_address_toggle <true | false>]
```

```
[-compute_redund_mem_address_toggle <true | false>]
[-report_redund_mem_data_toggle <true | false>]
[-compute_redund_mem_data_toggle <true | false>]
[-report_mem_light_sleep <true | false>]
[-compute_sleep_redund <true | false>]
[-report_stable_mem_access <true | false>]
[-report_redund_reset <true | false>]
[-report_clock_gating <true | false>]
[-report_mem_gating <true | false>]
[-report_data_gating <true | false>] [-report_rows_limit <rows>]
[-pl_db_gen_type <all | limited | skip>]
[-report_early_design_checks <true | false>]
```

-report_user_enabled_flops <true | false>

Determines whether to generate the user-enabled flops report. When set to `false`, it will disable the generation of the user-enabled flops report. Possible values are `true` and `false`. Default is `true`.

-report_blcg <true | false>

Determines whether to generate the block level clock gating report. When set to `false`, it will disable the generation of the block level clock gating report. Possible values are `true` and `false`. Default is `true`.

-blcg_flop_ratio_threshold <*threshold*>

Block level clock gating will report the gating opportunities for a hierarchical instance only if the ratio (number of flops gated by a given signal) / (total number of flops in that hierarchy) is greater than the value specified by this option. By default, the value is `0.9` and can be set within the range `0-1.0`

-report_sr2cb <true | false>

Determines whether to generate the shift register to circular buffer report. When set to `false`, it will disable the generation of the shift register to circular buffer report. Possible values are `true` and `false`. Default is `true`.

-report_access_profile <true | false>

Determines whether to generate the access profile report. When set to `false`, it will disable the generation of the access profile report. Possible values are `true` and `false`. Default is `true`.

-ap_mem_addr_access_threshold <*threshold*>

Memory Access Report for a given memory enable port will generate those addresses if the ratio of (read + write operations on that address)`*100` / (total operations on that memory) is greater than or equal to the value specified by this option. By default, the value is `10.0`.

-compute_access_profile_redund <true | false>

Determines whether the qwave information for the access profile redundancy will be generated. When set to `false`, the qwave information will not be generated. Possible values are `true` and `false`. Default is `true`.

-report_bypass_potential <true | false>

Determines whether to generate the bypass potential report. When set to `false`, it will disable the generation of the bypass potential report. Possible values are `true` and `false`. Default is `true`.

-compute_bypass_redund <true | false>

Determines whether the qwave information for the bypass redundancy will be generated. When set to `false`, the qwave information will not be generated. Possible values are `true` and `false`. Default is `true`.

-report_redundant_mux <true | false>

Determines whether to generate the redundant mux activity report. When set to `false`, it will disable the generation of the redundant mux activity report. Possible values are `true` and `false`. Default is `true`.

-compute_mux_redund <true | false>

Determines whether the qwave information for the mux redundancy will be generated. When set to `false`, the qwave information will not be generated. Possible values are `true` and `false`. Default is `true`.

-report_high_togg_signals <true | false>

Determines whether to generate the high toggling signals report. When set to `false`, it will disable the generation of the high toggling signals report. Possible values are `true` and `false`. Default is `true`.

-report_clk_toggle_data_stable <true | false>

Determines whether to generate the clock toggle data stable report. When set to `false`, it will disable the generation of the clock toggle data stable report. Possible values are `true` and `false`. Default is `true`.

-compute_clk_toggle_data_stable_redund <true | false>

Determines whether the qwave information for the clock toggle data stable redundancy will be generated. When set to `false`, the qwave information will not be generated. Possible values are `true` and `false`. Default is `true`.

-report_redund_mem_address_toggle <true | false>

Determines whether to generate the redundant memory address toggle report. When set to `false`, it will disable the generation of the redundant memory address toggle report. Possible values are `true` and `false`. Default is `true`.

-compute_redund_mem_address_toggle <true | false>

Determines whether the qwave information for the memory address toggle redundancy will be generated. When set to `false`, the qwave information will not be generated. Possible values are `true` and `false`. Default is `true`.

-report_redund_mem_data_toggle <true | false>

Determines whether to generate the redundant memory data toggle report. When set to `false`, it will disable the generation of the redundant memory data toggle report. Possible values are `true` and `false`. Default is `true`.

-compute_redund_mem_data_toggle <true | false>

> Determines whether the qwave information for the memory data toggle redundancy will be generated. When set to `false`, the qwave information will not be generated. Possible values are `true` and `false`. Default is `true`.

-report_mem_light_sleep <true | false>

> Determines whether to generate the memory light sleep report. When set to `false`, it will disable the generation of the memory light sleep report. Possible values are `true` and `false`. Default is `true`.

-compute_sleep_redund <true | false>

> Determines whether the qwave information for the memory light sleep redundancy will be generated. When set to `false`, the qwave information will not be generated. Possible values are `true` and `false`. Default is `true`.

-report_stable_mem_access <true | false>

> Determines whether to generate the stable memory access report. When set to `false`, it will disable the generation of the stable memory access report. Possible values are `true` and `false`. Default is `true`.

-report_redund_reset <true | false>

> Determines whether to generate the redundant reset flops report. When set to `false`, it will disable the generation of the redundant reset flops report. Possible values are `true` and `false`. Default is `true`.

-report_clock_gating <true | false>

> Determines whether to generate the clock gating report. When set to `false`, it will disable the generation of the clock gating report. Possible values are `true` and `false`. Default is `true`.

-report_mem_gating <true | false>

> Determines whether to generate the memory gating report. When set to `false`, it will disable the generation of the memory gating report. Possible values are `true` and `false`. Default is `true`.

-report_data_gating <true | false>

> Determines whether to generate the data gating report. When set to `false`, it will disable the generation of the data gating report. Possible values are `true` and `false`. Default is `true`.

-report_rows_limit <*rows*>

> Specifies the maximum number of rows in which the reports must be generated. Possible values are `0` to `10,000`. Default is `10,000`.

-pl_db_gen_type <all | limited | skip>

> This option controls the generation of waveform information. Possible values are `all`, `limited`, and `skip`. Default is `limited`.

`all`: waveform information is generated for all the design signals

`limited`: waveform information is generated only for the signals that are available in PowerPro reports

`skip`: waveform information is not generated for any design signal

-report_early_design_checks <true | false>

Determines whether to generate the early design check reports. When set to `false`, it disables the generation of all the early design check reports. Possible values are `true` and `false`. Default is `true`.

Related Commands

report_html_format, show_analyzer, generate_guidance

Usage

Configures the behavior of guidance reports.

Examples

Example 1: Disables the high toggling signals report and set the rows limit.

```
config_guidance_reports -report_high_togg_signals 0 -report_rows_limit
10
```

Example 2: Disables the computation of the clock toggle data stable redundancy.

```
config_guidance_reports -compute_clk_toggle_data_stable_redund no
```

Example 3: Disables the generation of the block level clock gating, shift register to circular buffer, and redundant reset flops reports.

```
config_guidance_reports -report_blcg no -report_sr2cb no -
report_redund_reset no
```

Example 4: Disables the generation of the redundant memory address toggle report and computation of bypass redundancy, and sets the flop ratio threshold as `0.6` for block level clock gating report.

```
config_guidance_reports -report_redund_mem_address_toggle no
-compute_bypass_redund no -blcg_flop_ratio_threshold 0.6
```

Example 5: Disables the generation of the memory gating report and data gating report.

```
config_guidance_reports -report_mem_gating no -report_data_gating no
```

config_metrics

config_metrics — Configures the behaviour of metrics computation and their detailed reporting in a powerpro run. This command can be run at any stage.

Synopsis

```
config_metrics [-set_filter <filter_file> ] [ -reset_filter ]
```

-set_filter <*filter_file*>

> Specifies the filters provided in filter file for all the metrics of a specific configuration in the run.
>
> The filter file must be in the CSV form. The format should be:
>
> *<metric_name>, <filter_string>*

-reset_filter

> Specifies to reset all global filters passed into the run.

Related Commands

> report_format, show_analyzer

Usage

Configure the metrics for reporting and generating dashboard values.

Example

Example 1: Displays the report on PowerPro command line interface.

```
config_metrics -set_filter filter.csv
```

---

config_multicore_setup

config_multicore_setup — Configures the multi-core setting required for the multi-core PowerPro flow. The command cannot be run after the `report_power` command.

Synopsis

```
config_multicore_setup
[ -host_setup_configuration_pacompile <LSF command> ]
[ -ppro_machine_list_pacompile <file_path> ]
[ -max_remote_process_usage_pacompile <count> ]
[ -max_local_process_usage_pacompile <count> ]
[ -host_setup_configuration_pacompute <LSF command> ]
[ -ppro_machine_list_pacompute <file_path> ]
[ -max_remote_process_usage_pacompute <count> ]
[ -max_local_process_usage_pacompute <count> ]
[ -host_setup_configuration_dp <LSF command>]
[ -ppro_machine_list_dp <file_path>]
[ -max_remote_process_usage_dp <count>]
[ -max_local_process_usage_dp <count>]
[ -design_partition_count <count>]
[ -host_setup_configuration_slecpro <LSF command>]
[ -max_remote_process_usage_slecpro <count>]
[ -max_local_process_usage_slecpro <count>]
[ -num_parallel_solver_jobs_slecpro <count>]
```

-host_setup_configuration_pacompile <*LSF command*>

Specifies the network-specific settings that must be provided by the various options of the LSF (Load Sharing Facility) command, which is required to run the pseudo-synthesis process on the remote machine. Default is `empty`. The option –`max_remote_process_usage_pacompile` must be specified with this option.

-ppro_machine_list_pacompile <*file_path*>

Specifies the path to the file that contains the list of machines that are used to run the pseudo-synthesis process in parallel on the remote machines. Each line in the file must include a machine name that has password-less access with ssh from the host machine. If multiple cores of a machine are required, add multiple lines with the same machine name in the file. Blank lines are not allowed in the file. The option -`max_remote_process_usage_pacompile` must be specified with this option.

-max_remote_process_usage_pacompile <*count*>

Specifies the maximum number of parallel processes (per partition in case of design partition flow) that must be used for pseudo-synthesis on the remote machine. Possible values are greater than or equal to `zero` (0). Default is `0`. If value is greater than `0`, it requires at least one of –`host_setup_configuration_pacompile` or -`ppro_machine_list_pacompile`. The option `max_local_process_usage_pacompile` must be specified with this option.

-max_local_process_usage_pacompile <*count*>

Specifies the maximum number of parallel processes (per partition in case of design partition flow) that must be used for pseudo-synthesis on the local machine. Possible values are greater than or equal to `zero` (`0`). Default is `4`. The option `max_remote_process_usage_pacompile` must be specified with this option.

-host_setup_configuration_pacompute <*LSF command*>

Specifies the network-specific settings that must be provided by the various options of the LSF (Load Sharing Facility) command, that is required to run the power analysis process on the remote machine. Default is `empty`. The option –`max_remote_process_usage_pacompute` must be specified with this option.

-ppro_machine_list_pacompute <*file_path*>

Specifies the path to the file that contains the list of machines that are used to run the power analysis process in parallel on the remote machines. Each line in the file must include a machine name that has password-less access with ssh from the host machine. If multiple cores of a machine are required, add multiple lines with the same machine name in the file. Blank lines are not allowed in the file. The option –`max_remote_process_usage_pacompute` must be specified with this option.

-max_remote_process_usage_pacompute <*count*>

Specifies the maximum number of parallel processes (per partition in case of design partition flow) that must be used for power analysis on the remote machine. Possible values are `0` and above. Default is `0`. If value is greater than `0` then it requires at least one of –`host_setup_configuration_pacompute` or –`ppro_machine_list_pacompute`. The option `max_local_process_usage_pacompute` must be specified with this option.

-max_local_process_usage_pacompute <*count*>

Specifies the maximum number of parallel processes (per partition in case of design partition flow) that must be used for power analysis on the local machine. Possible values are `0` and above. Default is `4` except for PowerPro-Veloce flow. The option –`max_remote_process_usage_pacompute` must be specified with this option.

-host_setup_configuration_dp <*LSF command*>

Specifies the network-specific settings that must be provided by the various options of the LSF (Load Sharing Facility) command, that is required to run the partitions of the design partition flow on the remote machine. This option is supported only in Enhanced-PA flow. Default is `empty`. The option –`max_remote_process_usage_dp` must be specified with this option.

-ppro_machine_list_dp <*file_path*>

Specifies the path to the file that contains the list of machines that are used to run the partitions of the design partition flow on the remote machines. Each line in the file must include a machine name that has password-less access with ssh from the host machine. If multiple cores of a machine are required, add multiple lines with the same machine name in the file. Blank lines are not allowed in the file. This option is supported only in Enhanced-PA flow. The option –`max_remote_process_usage_dp` must be specified with this option.

-max_remote_process_usage_dp <*count*>

Specifies the maximum number of the partitions of the design partition flow that would be launched on the remote machine. This option is supported only in Enhanced-PA flow. Possible values are `0` and above. Default is `0`. If the value is greater than `0`, it requires at least one of –`host_setup_configuration_dp` or –

ppro_machine_list_dp options. The option -max_local_process_usage_dp must be specified with this option.

-max_local_process_usage_dp <*count*>

Specifies the maximum number of the partitions of the design partition flow that would be launched on the local machine. This option is supported only in Enhanced-PA flow. Possible values are 0 and above. Default is 0. The option –max_remote_process_usage_dp must be specified with this option. The recommended value of PowerPro-Veloce flow is atleast 1.

-design_partition_count <*count*>

Specifies the number of the partitions to be launched in the design partition flow. The total number of the partitions would be 1 more than the partition count specified. Default is 0.

-host_setup_configuration_slecpro <*LSF command*>

Specifies the network-specific settings that must be provided by the various options of the LSF (Load Sharing Facility) command, which is required to run the slecpro process on the remote machine. Default is empty. The option –max_remote_process_usage_slecpro must be specified with this option.

-max_remote_process_usage_slecpro <*count*>

Specifies the maximum number of parallel processes that must be used for slecpro on the remote machine. Possible values are greater than or equal to zero (0). Default is 1. If value is greater than 0, it requires –host_setup_configuration_slecpro. The option –max_local_process_usage_slecpro must be specified with this option.

-max_local_process_usage_slecpro <*count*>

Specifies the maximum number of parallel processes that must be used for slecpro on the local machine. Possible values are greater than or equal to zero (0). Default is 1. The option –max_remote_process_usage_slecpro must be specified with this option.

-num_parallel_solver_jobs_slecpro <*count*>

This will set the solver parallelization specific globals (enable_parallel_solverloop and solverloop_num_processes) for slec to verify Powerpro Observability based optimizations. To enable parallelization, its value should be greater than 1. Default is 0.

Related Commands

prototype_design, report_power

Usage

Configures the multi-core setting required for the enhanced power analysis flow.

Examples

Example 1: Specifies the local machines for pseudo-synthesis and power analysis to be 8 and 4 respectively.

```
config_multicore_setup -max_local_process_usage_pacompile 8 -
max_remote_process_usage_pacompile 0 -max_local_process_usage_pacompute
4 -max_remote_process_usage_pacompute 0
```

Example 2: Specifies the remote machines for pseudo-synthesis and power analysis to be 8 and 4 respectively, where remote machines are specified using LSF settings. Values for -host_setup_configuration_pacompile and -host_setup_configuration_pacompute options must be specified based on LSF settings.

```
config_multicore_setup \
    -host_setup_configuration_pacompile "..."
    -max_remote_process_usage_pacompile 8 \
    -max_local_process_usage_pacompile 0 \
    -host_setup_configuration_pacompile "..."
    -max_remote_process_usage_pacompute 4 \
    -max_local_process_usage_pacompute 0
```

Example 3: Specifies the remote machines for pseudo-synthesis and power analysis to be 8 and 4 respectively, where remote machines are specified using names in the files. Content of the file machine_list_pacompile.txt and machine_list_pacompute.txt should to specified based on available machines in the network.

```
config_multicore_setup \
    -ppro_machine_list_pacompile machine_list_pacompile.txt
    -max_remote_process_usage_pacompile 8 \
    -max_local_process_usage_pacompile 0 \
    -ppro_machine_list_pacompute machine_list_pacompute.txt
    -max_remote_process_usage_pacompute 4 \
    -max_local_process_usage_pacompute 0
```

Example 4: Specifies the number of partitions launched in the design partition flow to be 4. Partitions are distributed on local and remote machines to be 1 and 4 respectively, where remote machines are specified using LSF settings. Values for –host_setup_configuration_dp option must be specified based on LSF settings. This would result in total of 20 (5 partitions and 4 machines per partition) remote machines for power analysis and 40 (5 partitions and 8 machines per partition) remote machines for pseudo-synthesis.

```
config_multicore_setup \
    -design_partition_count 4
    -max_remote_process_usage_dp 4 \
    -max_local_process_usage_dp 1 \
    -host_setup_configuration_dp "..." \
    -ppro_machine_list_pacompile machine_list_pacompile.txt
    -max_remote_process_usage_pacompile 8 \
    -max_local_process_usage_pacompile 0 \
    -ppro_machine_list_pacompute machine_list_pacompute.txt
    -max_remote_process_usage_pacompute 4 \
    -max_local_process_usage_pacompute 0
```

Example 5: Specifies the number of partitions launched in the design partition flow to be `4`. Partitions are distributed on local and remote machines to be `1` and `4` respectively, where remote machines are specified using names in the file. Content of the file `machine_list_dp.txt` should to specified based on available machines in the network.

```
config_multicore_setup \
    -design_partition_count 4
    -max_remote_process_usage_dp 4 \
    -max_local_process_usage_dp 1 \
    -ppro_machine_list_dp machine_list_dp.txt
```

config_opt_expr

config_opt_expr — Sets the configuration parameters for the enable expressions and breaks down the enable expressions to generate simpler expressions.

Synopsis

```
config_opt_expr [ -obs | -stb ]
      [ -max_expr_comb_operators <number_of_instances> ]
      [ -max_expr_comb_depth <combinational_depth> ]
      [ -max_expr_support_signals <number_of_support_signals> ]
      [ -max_expr_hier_depth <number_of_hierarchical_boundaries> ]
      [ -max_expr_seq_elements <number_of_sequential_elements> ]
      [ -max_expr_area <maximum_expression_area> ]
```

-obs | -stb

Specifies whether the configuration parameters must be applied to observability-based or stability-based sequential optimization commands. By default, the configuration parameters are applied to both observability-based and stability-based sequential optimization commands.

-max_expr_comb_operators <*number_of_instances*>

Specifies the maximum number of instances in the enable expression.

-max_expr_comb_depth <*combinational_depth*>

Specifies the maximum combinational depth of the enable expression.

-max_expr_support_signals <*number_of_support_signals*>

Specifies the maximum number of support signals in the enable expression.

- max_expr_hier_depth <*number_of_hierarchical_boundaries*>

Specifies the number of hierarchical boundaries to be crossed during routing of support signals.

-max_expr_seq_elements <*number_of_sequential_elements*>

Specifies the maximum number of sequential elements in an enable expression. This option can be used only with the `-stb` option.

-max_expr_area <*maximum_expression_area*>

Specifies the maximum area of the enable expression.

Related Commands

insert_observability_logic, insert_stability_logic, insert_mem_observability_gating, insert_mem_stability_gating

Usage

This command is used to set the configuration parameters for the enable expressions. The configuration parameters must be set before performing optimizations.

Examples

Example 1: Illustrates how the observability-based and stability-based engines are used to generate expressions with a maximum of two operators.

```
    build_design input.v
    prototype_design
    config_opt_expr -max_expr_comb_operators 2
    insert_observability_logic
```

Example 2: Illustrates how only the observability-based engine can generate expressions with a maximum of two operators.

```
build_design input.v
prototype_design
config_opt_expr -obs -max_expr_comb_operators 2
insert_observability_logic
```

Example 3: Illustrates how only the stability-based engine can generate expressions with a maximum of two operators.

```
build_design input.v
prototype_design
config_opt_expr -stb -max_expr_comb_operators 2
insert_stability_logic
```

Example 4: Directs the memory-based observability and stability engines to generate expressions with a maximum of two operators.

```
build_design input.v
prototype_design
config_opt_expr -max_expr_comb_operators 2
insert_mem_observability_gating
insert_mem_stability_gating
```

config_reset_spec

config_reset_spec — Allows the user to specify the name and location of the reset module and its instances, as well as the names of the synchronizer modules. Note that this command can only be called after `prototype_design` or before calling any sequential optimization command.

The rationale behind introducing this command follows. PowerPro generates a `ppro_reset_inst` instance that is used to control the `aol`, `mode_override`, and `set_powerpro_reset` signals defined in the PowerPro setup. By default, this instance is placed at the top-level with the default name `design_top_module_name_powerpro_reset_inst`. However, to maintain the readability of the code, it can be placed at a specific location and with a specific name by using the `config_reset_spec` command.

Synopsis

```
config_reset_spec [ -reset_instance <inst_name>]
    [ -reset_hier <hier_name>] [ -location <name_of_instance>]
    [ -reset_pos_sync <positive_reset_synchronizer_name>]
    [ -reset_neg_sync <negative_reset_synchronizer_name>]
    [ -data_sync <data_synchronizer_name>]
    [ -create_short_sync_inst_name]
```

-reset_instance *`<inst_name>`*

Specifies the name of the reset instance. If this option is not specified, the default name `design_top_module_name_powerpro_reset_inst` is used.

-reset_hier *`<hier_name>`*

Specifies the name of the reset module. If this option is not specified, the default name `design_top_module_name_powerpro_reset_mod` is used.

-location *<name_of_instance>*

Specifies the hierarchical name of the instance in the design where the reset instance is to be instantiated. For top modules, location name is top module name. When this option is not specified, the reset instance is instantiated in the top module.

-reset_pos_sync *<positive_reset_synchronizer_name>*

Specifies the name of the positive reset synchronizer module. If this option is not specified, the default name `calypto_sync_%s_pos` is used, where `%s` is name of the top module.

-reset_neg_sync *<negative_reset_synchronizer_name>*

Specifies the name of the negative reset synchronizer module. If this option is not specified, the default name `calypto_sync_%s_neg` is used, where `%s` is name of the top module.

-data_sync *<data_synchronizer_name>*

Specifies the name of the data synchronizer module. If this option is not specified, the default name `calypto_sync_%s_mod` is used, where `%s` is the name of the top module.

-create_short_sync_inst_name

Creates short names for the synchronizer instances.

Related Commands

set_powerpro_reset

Usage

Using the `config_reset_spec` command, the user can specify the name and location of the reset module and its instances, as well as the names of the synchronizer modules.

Examples

Example 1: Sets the reset instance name to `pp_rst_inst` and the reset module name to `pp_rst_mod`. The name of the positive reset synchronizer module is set to `pos_rst_sync`, the negative reset synchronizer to `neg_rst_sync` and the data synchronizer module to `d_sync`. The reset instance is created in the instance `correlator.ecc_inst`. Short names for the synchronizer instances are also created.

```
config_reset_spec  -reset_instance pp_rst_inst
     -reset_hier pp_rst_mod    -location correlator.ecc_inst
     -reset_pos_sync pos_rst_sync    -reset_neg_sync neg_rst_sync
     -data_sync d_sync   -create_short_sync_inst_name
```

config_write_rtl

config_write_rtl — Configures the output generated by `write_rtl`. It is recommended that the `config_write_rtl` command be called before `build_design`. Also note that the `config_write_rtl` command cannot be called after any enable optimization command.

Synopsis

```
config_write_rtl [ -prefix_in_all_new_hierarchies <string>]
    [-observability_logic_instance_comment <string>]
    [-observability_logic_instance_naming_scheme <format -string>]
    [-observability_logic_module_naming_scheme <format -string>]
    [-stability_logic_instance_comment <string>]
    [-const_stability_logic_instance_naming_scheme <format -string>]
    [-const_stability_logic_module_naming_scheme <format -string>]
    [-sym_stability_logic_instance_naming_scheme <format -string>]
    [-sym_stability_logic_module_naming_scheme <format -string>]
    [-mem_observability_logic_module_naming_scheme <format -string>]
    [-mem_observability_logic_instance_naming_scheme <format -string>]
    [-mem_const_stability_logic_module_naming_scheme <format -string>]
    [-mem_const_stability_logic_instance_naming_scheme <format-string>]
    [-mem_sym_stability_logic_module_naming_scheme <format -string>]
    [-mem_sym_stability_logic_instance_naming_scheme <format -string>]
    [-mem_light_sleep_logic_module_naming_scheme <format -string>]
    [-mem_light_sleep_logic_instance_naming_scheme <format -string>]
    [-rerouted_signal_comment <string>]
    [-rerouted_signal_naming_scheme <format -string>]
    [-added_port_argument_comment <string>]
    [-enlogic_file_naming_scheme <string>]
    [-uniquified_module_comment <string>]
    [-uniquified_module_naming_scheme <format -string>]
    [-rename_uniquified_module_in_comments yes | no ]
    [-assignment_guarding_comment <string>]
    [-assignment_guarding_signal_naming_scheme <format -string>]
    [-rename_all_modified_modules yes | no ]
    [-reuse_existing_module yes | no ]
    [-error_if_cannot_separate_uniquified_module_files yes | no]
    [-separate_uniquified_module_files yes | no ]
    [-force_separate_uniquified_module_files yes | no ]
    [-verilog_ifdef_guard <string>]
    [-preprocess_verilog yes | no ]
    [-expand_include_outside_module yes | no ]
    [-rewrite_verilog_parameters yes | no ]
    [-vhdl_new_logic_library_name <string>]
    [-vhdl_new_logic_package_name <string>]
    [-vhdl_tc_package_name <string>]
    [-catapult yes | no ]
    [-generate_X_aware_enable yes | no ]
    [-allow_ternary_patching yes | no ]
    [-make_ieee_1164_visible_in_declaration yes | no ]
    [-enable_logic_style merge | inline | separate ]
    [-v95_compliant yes | no ]   [ -use_iterative_write_rtl yes | no]
    [-convert_logic_to_wire yes | no]
    [-patch_within_generate yes | no ] [-iwr_pre_script <string> ]
    [-create_directory_for_design_files yes | no ]
    [-allow_vpp_for_ifdef_directives yes | no]
```

-prefix_in_all_new_hierarchies *<string>*

Specifies that a prefix be added to the names of all newly created hierarchies.

-observability_logic_instance_comment *<string>*

Specifies the comment that gets written when an observability logic instance is added by `write_rtl`. Default: `PowerPro -CG`.

-observability_logic_instance_naming_scheme *<format -string>*

Specifies the naming scheme to be used for any new observability logic instance added by `write_rtl`. By default, the naming convention used is `inst_cg_obs_%s` where, `%s` is the name of the parent container module.

-observability_logic_module_naming_scheme *<format -string>*

Specifies the naming scheme to be used for new observability logic modules added by `write_rtl`. By default, the naming convention used is `cg_obs_%s` where `%s` is the name of the parent container module.

-stability_logic_instance_comment *<string>*

Specifies the comment that gets written when a stability logic instance is added by `write_rtl`. Default: `PowerPro -CG`.

-const_stability_logic_instance_naming_scheme *<format -string>*

Specifies the naming scheme to be used for new constant stability logic instances added by `write_rtl`. By default, the naming convention used is `inst_cg_const_stb_%s` where, `%s` is the name of the parent container module.

-const_stability_logic_module_naming_scheme *<format -string>*

Specifies the naming scheme to be used for new constant stability logic modules added by `write_rtl`. By default, the naming convention used is `cg_const_stb_%s` where, `%s` is the name of the parent container module.

-sym_stability_logic_instance_naming_scheme *<format -string>*

Specifies the naming scheme to be used for new symbolic stability logic instances added by `write_rtl`. By default, the naming convention used is `inst_cg_sym_stb_%s` where, `%s` is the name of the parent container module.

-sym_stability_logic_module_naming_scheme *<format -string>*

Specifies the naming scheme to be used for new constant stability logic modules added by `write_rtl`. By default, the naming convention used is `cg_sym_stb_%s` where, `%s` is the name of the parent container module.

-mem_observability_logic_module_naming_scheme *<format -string>*

Specifies the naming scheme to be used for new memory observability logic modules added by `write_rtl`. By default, the naming convention used is `mg_obs_%s` where, `%s` is the name of the parent container module.

-mem_observability_logic_instance_naming_scheme *<format -string>*

Specifies the naming scheme to be used for new memory observability logic instances added by `write_rtl`. By default, the naming convention used is `inst_mg_obs_%s` where, `%s` is the name of the parent container module.

-mem_const_stability_logic_module_naming_scheme *<format -string>*

Specifies the naming scheme to be used for new memory constant stability logic modules added by `write_rtl`. By default, the naming convention used is `mg_const_stb_%s` where, `%s` is the name of the parent container module.

-mem_const_stability_logic_instance_naming_scheme *<format -string>*

Specifies the naming scheme to be used for new memory constant stability logic instances added by `write_rtl`. By default, the naming convention used is `inst_mg_const_stb_%s` where, `%s` is the name of the parent container module.

-mem_sym_stability_logic_module_naming_scheme *<format -string>*

Specifies the naming scheme to be used for new memory symbolic stability logic modules added by `write_rtl`. By default, the naming convention used is `mg_sym_stb_%s` where, `%s` is the name of the parent container module.

-mem_sym_stability_logic_instance_naming_scheme *<format -string>*

Specifies the naming scheme to be used for new memory symbolic stability logic instances added by `write_rtl`. By default, the naming convention used is `inst_mg_sym_stb_%s` where, `%s` is the name of the parent container module.

-mem_light_sleep_logic_module_naming_scheme *<format -string>*

Specifies the naming scheme to be used for new light sleep logic modules added by `write_rtl`. By default, the naming convention used is `mg_ls_%s` where, `%s` is the name of the parent container module.

-mem_light_sleep_logic_instance_naming_scheme *<format -string>*

Specifies the naming scheme to be used for new light sleep logic instances added by `write_rtl`. By default, the naming convention used is `inst_mg_ls_%s` where, `%s` is the name of the parent container module.

-rerouted_signal_comment *<string>*

Specifies the comment to be added when a new signal is routed by `write_rtl`. Default: `PowerPro -CG`.

-rerouted_signal_naming_scheme *<format -string>*

Specifies the naming scheme to be applied to new signals that are routed across hierarchies. Default is `%s` where, `%s` is the original signal name or already uniquified name of the original signal. Signal names are uniquified by appending a digit to the signal name i.e., `%s_<digit>`. For example, `sig_0, sig_1`. Uniquified signal names help where a signal or port by the same name already exists in the design.

-uniquified_module_comment *<string>*

Specifies the comment to be added when a module is uniquified by `write_rtl`. Default: `PowerPro -CG`.

-added_port_argument_comment *<string>*

Specifies the comment to be added when `write_rtl` adds a new port argument to a module. Default: `PowerPro -CG`.

-enlogic_file_naming_scheme *<string>*

Specifies the naming convention for the file used by `write_rtl` for writing definitions of newly created hierarchies. Note that this option is ignored when the `-enable_logic_style` option is set to `inline` or `separate`. Default: `enLogic`.

-uniquified_module_naming_scheme *<format -string>*

Specifies the naming scheme to be applied to modules that have been uniquified by `write_rtl`. Default is `%s_PP_` where, `%s` is the name of the original module. In case module names are uniquified, a digit is appended to the generated name. For example, the module `mod` would be renamed as `mod_PP_0` or `mod_PP_1` and so on.

-rename_uniquified_module_in_comments yes | no

Specifies that PowerPro replace all occurrences of the renamed module name in comment strings, from start to end of the module definition. Default: `yes`.

-assignment_guarding_comment *<string>*

Specifies that a comment be added when `write_rtl` guards an existing assignment with a PowerPro generated enable. Default: `PowerPro-CG`.

-assignment_guarding_signal_naming_scheme *<format -string>*

Specifies the naming scheme to be used for the signal that `write_rtl` uses to guard an existing assignment. By default, it is `%s_en` where, `%s` is the name of the left -hand -side variable being guarded.

-generate_X_aware_enable yes | no

Specifies that the enables generated by the CG hierarchies be `X` (don't care) aware. This would imply that any `X` value on the enable will be converted to `1`. As a result, the flop will not be gated. Such a conversion will take place only during simulation. Default: `yes`.

-allow_ternary_patching yes | no

This option controls how PowerPro patches the RTL for the gated synchronous flops that are modeled via ternary operator. When this option is enabled, the `write_rtl` command creates a patched RTL which includes ternary gating expressions for ternary synchronous flops. By default, this option is set to `yes`.

Note: This option will be ineffective if synchronous reset aware flow is not enabled during optimization.

Consider the following example.

```
  always@(posedge clk)
     flop <= sync_rst ? 0 : d;
// When the option is disabled, patching will be
    always@(posedge clk)
       if(pp_en)
          flop <= sync_rst ? 0 : d;
....
....
// When the option is enabled, patching will be
    always@(posedge clk)
       flop <= sync_rst ? 0:(pp_en ? d : flop);
```

-rename_all_modified_modules yes | no

This option has been deprecated.

Use the `-reuse_existing_module` option, instead.

-reuse_existing_module yes | no

When set to `yes`, PowerPro inserts gating logic in the original module itself. When set to `no`, PowerPro leaves the original module as-is and creates new modules by adding the suffix `_PP_<digit>` to the the original module name. For example, if the original module is named `mod`, the new module would be named `mod_PP_0`. By default, this option is set to `yes`.

-error_if_cannot_separate_uniquified_module_files yes | no

This option is no longer supported.

-separate_uniquified_module_files yes | no

When set to `yes`, this option writes uniquified modules to individual files. The generated file contains a renamed and optimized version of the original module. By default, this option is set to `no`.

-force_separate_uniquified_module_files yes | no

This option has been deprecated. Use the -separate_uniquified_module_files option, instead.

-verilog_ifdef_guard *<string>*

This option has been deprecated.

-preprocess_verilog yes | no

Specifies that the preprocessor be invoked during `build_design` for Verilog designs. By default, this option is set to `no`, and `build_design` does not call the preprocessor. However, if this option is set to `yes`, the preprocessor first flattens the Verilog files and then passes the preprocessed files to `build_design`. Since the output of this option is passed on to the `build_design` command, it must always be called before `build_design`.

Preprocessing is typically required in the following cases:

1. When the module name is not hardcoded but instead substituted by `macro replacement.

2. When the module body has some of its declarations and/or statements inlined by `include of another file.

3. When the always block statements are not hardcoded but instead substituted by `macro(clk,din,qout) expansion.

-expand_include_outside_module yes | no

Specifies that `` `include `` files, which are outside the scope of the module, be excluded from flattening when the `-preprocess_verilog` option is set to `yes`.

-rewrite_verilog_parameters yes | no

Verilog module instantiation parameters are retained when an instance is uniquified. Use this option to rewrite the default parameter values to match those of the instantiation override statement.

-vhdl_new_logic_library_name *<string>*

By default, `write_rtl` puts new logic entities and supporting packages into a library called `calypto_lib`. This option allows you to provide a different name. When providing a different name, be careful not to use a name like `work`. While this is allowed, it is a special VHDL keyword and can cause adverse side -effects on existing package dependencies.

-vhdl_new_logic_package_name *<string>*

By default, `write_rtl` writes new logic entities into a package called `calypto_package`. This option allows you to provide a different name for the package.

-vhdl_tc_package_name *<string>*

By default, `write_rtl` writes type -conversion functions into a package called `powerpro_tc_package`. This option allows you to provide a different name for the package.

-catapult yes | no

This option is no longer supported.

-make_ieee_1164_visible_in_declaration yes | no

By default, any new data declaration added to the patched RTL, is prefixed `ieee.std_logic_1164` in the type. Setting this option to `no` removes this prefix in the data declaration and prints the `use` clause for the `std_logic_1164` package additionally.

-enable_logic_style merge | inline | separate

Specifies the way the enable logic (CG/MG/LS hierarchies definition) is written. By default, the `merge` option is applied.

Use `merge` to instruct `write_rtl` to write out all enable logic hierarchies into a single new file - `enLogic.v`.

Use `inline` to instruct `write_rtl` to write the CG/MG/LS hierarchy definitions to the corresponding parent module files.

Use `separate` to instruct `write_rtl` to create a new file for each CG/MG/LS hierarchy. The naming convention applied to the file is `<hierarchy_name>.<extension_of_parent_module>`. For example, if the name of the hierarchy is `cg_obs_abc` and the name of the parent module file is `abc.v`, the resulting file will be named `cg_obs_abc.v`.

Note that the options `inline` and `separate` are currently supported only for Verilog and System Verilog Designs. In case these options are used with VHDL or Mixed designs, the default behavior will be applied i.e., a single `enLogic.v` file will be generated.

-v95_compliant yes | no

When set to yes, `write_rtl` prints its output in compliance with the Verilog 95 standard. By default, this option is set to `no`.

-use_iterative_write_rtl yes | no

Specifies that PowerPro analyze and clean -up the generated RTL to remove gating opportunities that could cause compilation issues. It is recommended that this option be used if syntactical errors are encountered while compiling the generated RTL

-convert_logic_to_wire yes | no

Decides what to specify in the new declaration if the data type of the signal is `logic`. When set to `yes`, the `write_rtl` command specifies the declaration as `wire <new_signal_name>`. When set to `no`, the declaration is specified as `logic <new_signal_name>`. Default: `No`.

-patch_within_generate yes | no

Determines whether the new assignment statements must be written inside or outside the generate block. Possible values are `yes` and `no`. Default is `no`.

When set to `yes`, the new assignment statements are written inside the generate block. When set to `no`, the new assignment statements are written outside the generate block. Note that this option can be used only with verilog designs and for the generate blocks that are not followed by a `begin` keyword. The above statement applies only when a `generate` block is the only statement in the module.

-iwr_pre_script <*`string`*>

Specifies the settings that must be passed for syntax checking during the processing of Iterative Write RTL (IWR). This option must be used only when the –use_iterative_write_rtl option is enabled.

For example, during the processing of IWR, there are warnings of the type [VLOG-IED]. To convert such warnings into an error, we can use the setting `set_verbosity -error VLOG-IED`. The erroneous moves will then be de-committed by the Iterative Write RTL (IWR).

-create_directory_for_design_files yes | no

Creates a directory that contains only the design files. Possible values are `yes` and `no`. Default is `no`.

-allow_vpp_for_ifdef_directives yes | no

When set to `yes`, the predefined compiler directives such as ifdef, ifndef, elsif, else, and endif that are used to control the compilation of a Verilog description, are removed from the Verilog pre-processed file. The inactive blocks are also removed. Possible values are `yes` and `no`. Default is `no`.

Note: This option can be used only if the option `-preprocess_verilog` is enabled.

Related Commands

write_rtl

Usage

The `config_write_rtl` command is used to configure the output of `write_rtl`. Using this command, appropriate comments, suffixes or prefixes can be added to the names of signals added by PowerPro. Names of files created by `write_rtl` can also be controlled using this command.

Examples

Example 1: Appends the suffix `_EN` to the output enable of all PowerPro -added hierarchies.

```
config_write_rtl -assignment_guarding_signal_naming_scheme %s_EN
```

Example 2: The following example shows how to patch the instances `f1` and `f2`, of the module `mod1`.

Consider the following example.

```
//File: top.v

`include defines.v
 module mod1( `include "ports.v" )
 ...
endmodule

module top
   mod1 f1 ( ..);
   mod1 f2 (..);
endmodule.

// File: ports.v
sff,in, clk,reset

// File: defines.v
`define SIZE
`define DELAY
```

To ensure the instances f1 and f2 can be correctly patched, specify the following command:

```
config_write_rtl  -preprocess_verilog yes
```

The resulting patched RTL is shown below:

```
//File: top.v
`define SIZE
`define DELAY

module mod1(sff,in, clk,reset, pp_enable1)
...
endmodule

module mod1_PP_0(sff,in, clk,reset, pp_enable2,pp_enable3)
...
endmodule

module top
    mod1 f1 ( .., pp_enable1);
    mod1_PP_0 f2 (.., pp_enable2, pp_enable3);
endmodule
```

Note: In the above example, if the `expand_include_outside_module` option is also set to `no`, the `INCLUDE statements outside the scope of the modules will not be expanded. In this case, the patched RTL would continue to show:

```
`include "defines.v"
```

instead of:

```
//File: top.v
`define SIZE
`define DELAY
```

create_black_box

create_black_box — Creates a black box around one or all instances of a module.

Synopsis

```
create_black_box  -module <module> |  -inst <instance>
```

-module *<module>*

Module to blackbox. Note that this option must be used before the `build_design` command is called.

-inst *<instance>*

Module instance to blackbox. This option supports the use of wildcards and regular expressions.

Note that this option can only be used after the `build_design` command has been called.

Related Commands

build_design, build_db

Usage

The **create_black_box** command removes the internal logic of a module while preserving its interface. Instances of the blackboxed module will not drive their output ports. The input ports of a blackboxed instance become pseudo primary output ports of the design, while the output ports of the instance become pseudo primary input ports of the design.

The `create_black_box` command must be used for large memories and for blocks which contain logic that cannot be synthesized. Such blocks must be blackboxed using the -`module` option before calling the `build_design` command.

Note that the `create_black_box` command supports the use of wildcards and regular expressions.

Examples

Example 1: Blackboxes all instances of the module `fast_mem`.

```
create_black_box -module fast_mem
```

Example 2: Blackboxes only the `fast_mem_bank` instance of the `fast_mem` module.

```
create_black_box -inst mem_unit.fast_mem_bank
```

create_clock

create_clock — Creates a named clock signal for analysis purposes, optionally specifying design port(s) as root(s) for the signal. Note that this command can only be called after `build_design`.

Synopsis

```
create_clock  -name <signal_name>  -period <period>
     [ -waveform <waveform_list>] [ <source_port>...]
```

-name *<signal_name>*

The domain name of the clock signal.

-period *<period>*

The amount of time, in nanoseconds, after which the signal repeats. Note that this value must be greater than `0`.

-waveform *<waveform_list>*

Tcl list of signal rise and fall event times (in nanoseconds). The list must contain an even number of event times. The even -numbered events are taken to be signal rise times, while the odd -numbered events are taken to be signal fall times. (The first entry in the list is event #0.) The event times must be increasing, and the last time specified must be strictly less than the defined period `-period`.

If no waveform list is specified then it is assumed to be { 0.0 *<period>*/2.0 }, which defines a signal with a rising edge at time 0.0 and a falling edge at time *<period>*/2.0 nanoseconds.

<source_port>...

The source ports for the signal. The valid source ports are primary input ports.

If source ports are given then this is considered a "real" clock. Otherwise, it is considered a "virtual" clock, with no sources.

Related Commands

report_clocks, set_powerpro_reset, set_clock_domain

Usage

The **create_clock** command creates an ideal clocking signal that can be used for analysis purposes, and with commands that require a clock signal.

A few points to note:

- All clock signals must, at a minimum, be given a unique name and a period after which the waveform repeats.
- Signals are defined to be `ideal`. In other words, it takes 0 time for a signal to rise or fall between the low signal and high signal voltage levels i.e., it is a repeating step function.
- Named clock signals are required for timing analysis commands like `set_clock_domain` and `set_powerpro_reset`.
- A waveform may optionally be defined in the form of a Tcl list. The list consists of increasing event times, alternating between signal rising and falling events, where the first event time always corresponds to a rising event.

There must be an even number of event times. Additionally, the difference between the maximum event time and the minimum event time must be less than the period.

- If a waveform is not explicitly defined, it is taken to be `{ 0.0 <period>/2.0 }` with a rising event at time 0.0 ns and a falling event at time *<period>*/2.0 ns. Note that time units are in nanoseconds(ns).
- A list of source ports(clock roots) may optionally be defined. If such roots are given, then the signal is taken to be that of a real clock. Otherwise, it is considered to be a virtual clock, with no sources.

- All clock roots must be either primary inputs or black box output ports. In particular, internal leaf nodes and internal hierarchical ports are not allowed.
- If a port is a clock root for multiple clock signals then earlier designations will be ignored and only the last -defined clock signal specifying that port as a clock root will be used (last one wins).
- If there are errors in the supplied parameter, `create_clock` reports an error and does not create a signal. The exception to this rule is specified clock roots. For such clock roots, the signal is created but bad names and illegal ports are flagged and automatically eliminated from the list. However, if bad names and illegal ports cause the list to be empty, clock signals are not created.

Clock Networks

Clock networks are defined using the `create_clock` command which creates the real clocks. Specifically, the transitive fanout of a specified clock root i.e., the clock root and all ports and edges through any combinational logic and clock -gating integrated cell arcs, up to (and including) the clock input ports of sequential elements (flip -flops, latches, and memories) becomes designated as the clock network.

This designation process occurs when clocking information is requested internally by the tool, and not when the `create_clock` command is processed. As a result, the log file might display clock network messages away from the initial `create_clock` messages.

Waveforms

While it is possible to define a variety of clock signal waveforms, caution should be exercised. Problematic waveforms can cause dramatic increases in analysis times or result in extremely tight timing constraints.

- Avoid creating waveforms with many time events. This multiplies the number of possible pairs of edges that must be taken into consideration when determining the most constraining launch -edge/capture -edge pair for a given pair of sequential elements. For example:

```
create_clock  -name A  -period 18  -waveform {1 2 3 5 7 11 13 17}
```

- Avoid specifying waveforms containing events where the greatest common divisor (GCD) of the event times is equal to the power of 10 of the least significant digit of those events. For example:

```
create_clock  -name B  -period 0.293826  -waveform {0.022937
0.193639}
```

The GCD of the period and either of the waveform edges is 0.000001. Any path which is constrained on one side by this clock and on the other side by another clock will very likely have a required arrival time at the end point of 0.000001 ns. There would also be the issue of available precision of the floats used to represent the timing data.

- If multiple clock signals are defined, it is better if these signals' event times have relatively significant GCDs. For example,

```
create_clock  -name C1  -period 199
create_clock  -name C2  -period 141
```

The GCD of `C1` and `C2` is 1, so any path constrained on one side by `C1` and on other side by `C2` will very likely have a required arrival time at the end point of 1 ns, which is very tight given the size of the clock periods.

Clock Roots and Networks

Specifying clock roots causes clock networks to be created. A clock network is considered to be an asynchronous part of the design, and not subject to timing constraints. Indeed, they help define timing constraints as follows:

- Sequential elements. Sequential elements, such as flip -flops and latches, whose clock ports are in the transitive fanout (TFO) of clock roots, become constrained by those roots' clock signals. Specifically, the data input and output ports of those elements will be constrained by the propagated clock signals.
- Clock gating. Combinational gates and special clock -gating integrated cells in the TFO of clock roots become part of clock networks. Specifically, all ports of such gates in the TFO of a clock root are designated as part of clock networks. All other ports are outside of any clock network, and any port -to -port connection between them and clock network ports are automatically cut, creating end points for their networks, possibly cutting structural combinational loops in the process.

Example

Example 1: Creates a virtual clock signal with the name `foo_clock` and period of 5.0 ns. This clock will have a default waveform of `{ 0.0 2.5 }`.

```
create_clock  -name foo_clock  -period 5.0
```

Example 2: Creates a real clock signal by the name of `foo_clock`, period 5.0 ns, and clock roots at `croot1` and `croot2`. This clock will have a default waveform of `{ 0.0 2.5 }`.

```
create_clock  -name foo_clock  -period 5.0 { croot1 croot2 }
```

Example 3: Creates a real clock signal by the name of `foo_clock`, period 5.0 ns, waveform `{ 0.5 3.0 }`, and clock roots at `croot1` and `croot2`.

```
create_clock  -name foo_clock  -period 5.0  -waveform { 0.5 3.0 }
{ croot1 croot2 }
```

Example 4: Assuming `clk` is a 2-bit primary input port, with different clock domains, `create_clock` creates the clock signals `my_clk1` and `my_clk2`.

```
create_clock  -name my_clk1  -period 1.0 clk[1]
create_clock  -name my_clk2  -period 1.2 clk[2]
```

Example 5: Assuming `clk` is a 2-bit primary input port lying in the same clock domain, this can be defined as a bit range vector to `create_clock` as follows:

```
create_clock  -name my_clk1  -period 1.0 clk[1:2]
```

Alternatively, it can also be defined as follows:

```
create_clock  -name my_clk1  -period 1.0 {clk[1] clk[2]}
```

Example 6: Overrides clock information for the individual bits of the multi-bit clock port `clk`. After the execution of these commands, clock domain `CLK1` will correspond to the 0th and 1st bit of the clock port `clk` while clock domain `CLK2` will correspond to the 2nd bit of the clock port `clk`.

```
create_clock  -name CLK1  -period 1 clk[0:2]
create_clock  -name CLK2  -period 2 clk[2]
```

create_generated_clock

create_generated_clock — Creates a named clock signal, specifying design port(s) as root(s) for the named clock signal. Note that this command can only be called after `build_design` and before calling `prototype_design`.

Synopsis

```
create_generated_clock [ -name <clock_name>] [-source <reference_pin>]
    [-divide_by <divide_factor>] [-multiply_by <multiply_factor>] [-
    invert]<port_or_pin_list>
```

-name *`<clock_name>`*

Specifies the name of the generated clock. In case a clock name is not specified or a clock by the same name already exists in the domain, PowerPro creates a generated clock by the name of `clk_sdc*`, where `*` is a number.

-source <*reference_pin*>

Specifies the reference pin in the design from which the clock waveform is to be derived.

-invert

Specifies that the generated clock waveform is inverted with respect to the reference clock.

-divide_by <*divide_factor*>

Specifies the frequency division factor with respect to the reference clock.

-multiply_by <*multiply_factor*>

Specifies the frequency multiplication factor with respect to the reference clock.

`<port_or_pin_list>`

Specifies the port or pin list on which the clock is to be generated.

Related Commands

create_clock

Usage

The **create_generated_clock** command creates an ideal clocking signal that can be used for analysis purposes, and with commands that require a clock signal. All clock signals must be given a unique name.

Examples

Example 1: Creates a generated clock by the name of `cgc1` on the port `clk_generator_inst.g_clk1`.

```
create_generated_clock -name cgc1 clk_generator_inst.g_clk1
```

Example 2: Creates a generated clock by the name of `CGC` on port `clk_generator2.g_clk` with division factor of `3` with respect to reference clock `clk`.

```
create_generated_clock -name CGC -source clk -invert -divide_by 3
clk_generator2.g_clk
```

Example 3: Creates a generated clock by the name of `CGC2` on port `clk_generator3.g_clk` with multiplication factor `2` with respect to reference clock `clk`.

```
create_generated_clock -name CGC2 -source clk -invert -multiply_by 2
clk_generator3.g_clk
```

create_mode

create_mode — Creates a new mode. Note that this command can only be called after `build _design` and before calling `prototype_design`.

Synopsis

```
create_mode  -name <mode_name>
    [ -ignore_transition ] [ -dont_create_override ]
```

-name *<mode_name>*

Name of the mode to be created.

-ignore_transition

Ensures that the design does not transition in and out of a mode while a process is being executed.

-dont_create_override

Ensures that the signals specified with *<mode_name>* are not used to create the override logic.

Related Commands

create_mode_constraint, set_mode_override

Usage

The `create_mode` command is used to create a new mode. Using this command, the user can set the design to a specific mode of operation. To set a signal as the constraining signal for a mode, use the `create_mode_constraint` command.

Example

Example 1: Creates a new mode `test`.

```
create_mode  -name test
```

Example 2: Creates a new mode `test` of the type `ignore_transition`.

```
create_mode  -name test  -ignore_transition
```

Example 3: Creates a new mode `MODE1` that ensures the signals `s1` and `s2` are not used to create override logic.

```
create_mode  -name MODE1   -dont_create_override
create_mode_constraint  -mode MODE1   -module TOP   -signal s1 \
         -value 1'b0
create_mode_constraint  -mode MODE1   -module TOP   -signal s2  \
         -value 1'b1
```

Note: If there are other mode constraints defined, signals associated with them would still be used to create the override logic. Consider the following example.

```
create_mode  -name MODE1  -dont_create_override
create_mode  -name MODE2
create_mode_constraint  -mode MODE1   -module TOP   -signal s1 \
      -value 1'b0
create_mode_constraint  -mode MODE1   -module TOP   -signal s2  \
      -value 1'b1
create_mode_constraint  -mode MODE2   -module TOP   -signal s3  \
      -value 1'b0
create_mode_constraint  -mode MODE2   -module TOP   -signal s4  \
      -value 1'b1
```

In this example, while the signals `s1` and `s2` will not be used to create the override logic, the signals `s3` and `s4` will be used.

create_mode_constraint

create_mode_constraint — Sets the specified signal as a constraining signal for the given mode.

Synopsis

```
create_mode_constraint -mode <mode_name> -module <module> | -instance
    -signal <signal>   -value <constant_value>
```

-mode *<mode_name>*

Name of the mode for which the constraint is to be set. If the mode does not exist, PowerPro exits with an error. To create a mode, use the `create_mode` command.

-module *<module>* | -instance

Specify `-module` to provide the name of the module the signal belongs to. The constraint is applied to the signal, specified by `-signal`, in all instances of this module.

Specify `-instance` to apply the constraint to the signal specified by `-signal`. This option expects a hierarchical name for *<signal>*.

-signal *<signal>*

Name of the signal on which the constraint is to be applied.

If `-instance` is specified then *<signal>* must be the hierarchical name of the signal. The constraint will be applied to the specified signal within the instance.

If the signal expression represents a slice or a bit of a signal, then those bits of the signal are constrained. The syntax for expressing the slices follows that of the language of the module in which the signal exists. For example to specify the signal, `out_sig`, in a Verilog module, specify `out_sig[1]` or `out_sig[5:3]`. Likewise, if the signal `out_sig` was in a VHDL module, specify `out_sig(1)`, `out_sig(5 downto 3)`.

-value *<constant_value>*

Constant value with which the signal is to be constrained. Value can be in decimal, binary, hexadecimal or octal form.

Related Commands

create_mode, set_case_analysis, set_mode_override

Usage

The `create_mode_constraint` sets a signal as the constraining signal for a given mode. Corresponding override logic is also created such that whenever any of the constraining signals of the given mode gets a value that differs from its constraining value, all CG/MG moves are overridden.

Example

Example 1: Constrains the signal `row[1:0]` with the value `2'b10` for all instances of module `two_tap`.

```
create_mode_constraint  -mode test  -module two_tap
     -signal row[1:0]  -value 2'b10
```

Example 2: Constrains the signal `new_inst.start` with the value `100`.

```
create_mode_constraint  -mode test  -instance
     -signal new_inst.start  -value 100
```

Example 3: Constrains the signal `out_sig(5 downto 1)` with the value `5'b11011` for all instances of module `mod_str`.

```
create_mode_constraint  -mode test  -module mod_str
     -signal {out_sig(5 downto 1)}  -value 5'b11011
```

---

define_memory_operation

define_memory_operation — Defines memory operations such as read and write. Information gathered through this command is used by PowerPro to generate power opt models required for sequential optimization of memories. Note that PowerPro does not verify the functional accuracy of the power opt models generated. It is important that you verify the information provided to get accurate power results. Note that this command can only be run after the `read_library` command and before running the `build_design` or `link_design` commands.

Synopsis

```
define_memory_operation -module < module_name>
    [-memory_type <1p|2p|1r1w>] { -read | -write | -sleep }
    [ -data_in <input_ports_expression> |
      -data_out <output_ports_expression> ]
    [ -clock_and_polarity <tuple(clock_expression, clock_polarity)> ]
    [ -enable <memory_enable_expression> ]
    [ -address <read_or_write_address> ]
    [ -when <when_condition> ]    [-write_mask <mask_expression>]
    [ -writethru ]    [ -sleep_mode_when <condition> ]
    [ -address_width <width>] [-word_width <width>]
```

-module *<module_name>*

Specifies the name of the module for which the operation is to be defined, where *<module_name>* is a technology cell previously read using the `read_library` command.

-memory_type *<1p|2p|1r1w>*

> Specifies the type of memory for which information is being provided. Legal values are: `1p`, `2p` and `1r1w`.

-read | -write | -sleep

> Specify `-read` to define a read operation for the memory. This option expects the `-data_out` option to be specified.
>
> Specify `-write` to define a write operation for the memory. This option expects the `-data_in` option to be specified.
>
> Specify `-sleep` to define a sleep operation for the memory. This option expects the `-sleep_mode_when` option to be specified.

-data_in *<input_ports_expression>* | -data_out *<output_ports_expression>*

> Specify `-data_in` to provide an expression consisting of input ports required by the write memory operation.
>
> Specify `-data_out` to provide an expression consisting of output ports required by the read memory operation.

-clock_and_polarity *<tuple(clock_expression, clock_polarity)>*

> Specifies a tuple corresponding to the clock pin expression and the clock polarity of the module. The clock polarity indicates whether the memory operation is to be triggered on a rising or falling transition of the specified clock pin. Legal values for polarity are `pos` and `neg`. Note that it is important to retain this order.

-address*<read_or_write_address>*

> Specifies the address to be read from or to be written to. This option must be specified for each read or write operation.

-enable*<memory_enable_expression>*

> Specifies the enable expression, comprising of pin states, required to trigger the operation on the memory. This option is required when memory type is `1p` or `2p`.

-when*<when_condition>*

> Specifies the condition that triggers a read or write operation on the memory.

-sleep_mode_when *<condition>*

> Specify `-sleep_mode_when` to describe the condition that will trigger the memory to go into light sleep.

-write_mask*<mask_expression>*

> Specifies the value of the write mask for the memory write data expression. This should be equivalent to the size of the write data expression. By default, bits are not masked and everything is written as-is to the memory.

-writethru

Specifies whether or not the memory has write thru functionality. Specifying this option ensures that data written to the data port, specified by the *<input_ports_expression>*, is read and available at the data output port, specified by *<output_ports_expression>*.

-address_width <*width*>

Specifies the width of the address bus of the memory. This option can be used if the liberty file (`.lib`) does not contain this information, and can be used either with the read or the write operation.

Note: If the liberty file (`.lib`) contains this information and the `define_memory_operation` command is used, the data from the `.lib` file is considered.

-word_width <*width*>

Specifies the width of a word of the memory. This option can be used if the liberty file (`.lib`) does not contain this information, and can be used either with the read or the write operation.

Note: If the liberty file (`.lib`) contains this information and the `define_memory_operation` command is used, the data from the `.lib` file is considered.

Related Commands

insert_mem_observability_gating, insert_mem_stability_gating, insert_me_sleep_gating, report_memory_power

Usage

The `define_memory_operation` command gathers information about clocks, enable conditions, mask conditions, address conditions, power-down conditions and behavior of memories in the design. PowerPro uses this information to generate power opt models.

Note that PowerPro does not verify the functional accuracy of the power opt models generated. It is assumed that the information provided is accurate. It is recommended that you verify the information provided as the power opt model used impacts the results generated.

For ease of reference, the syntax for the different memory operations is provided below.

Syntax for Read Memory Operations

```
define_memory_operation
     -read {-module <module_name>}
     {-memory_type <1p|2p|1r1w>}
     {-data_out <expression>}
     {-address <read_or_write_address>}
     {-clock_and_polarity <tuple(clock_expression, clock_polarity)>}
     {-enable <expression>}
     {-when <when_condition>}
```

Syntax for Write Memory Operations

```
define_memory_operation
     -write {-module <module_name>}
     {-memory_type <1p|2p|1r1w>}
     {-data_in <expression>}
     {-address <read_or_write_address>}
     {-clock_and_polarity <tuple(clock_expression, clock_polarity)>}
     {-enable <expression>}
     {-when <when_condition>}
     [-writethru] [-write_mask <mask_expression>]
```

Syntax for Sleep Memory Operations

```
define_memory_operation
     -sleep
     -sleep_mode_when <light_sleep_condition>
```

Examples

Example 1: Defines memory interface information for different operations of the 2p memory module `mem2pcell`.

```
define_memory_operation  -module mem2pcell -memory_type 2p -read
-clock_and_polarity {CLKA pos} -data_out QA -address RADRA -enable "MEA
& MEA1 & MEA2" -when "~WEA"

define_memory_operation -module mem2pcell -memory_type 2p -write
-clock_and_polarity {CLKA pos} -data_in WDATA -address WADRA -enable
"MEA & MEA1 & MEA2" -when "WEA" -write_mask "4{1'b0}"

define_memory_operation -module mem2pcell -memory_type 2p -read
-clock_and_polarity {CLKB pos} -data_out QB -address RADRB  -enable
"MEB & MEB1 & MEB2" -when "~WEB"

define_memory_operation  -module mem2pcell -memory_type 2p -write
-clock_and_polarity {CLKB pos}  -data_in WDATB -address WADRB -enable
"MEB & MEB1 & MEB2" -when "WEB" -write_mask "4{1'b1}"

define_memory_operation -module mem2pcell -sleep -sleep_mode_when LS
```

Example 2: Defines memory interface information for different operations of the 1p memory module `mem1pcell`.

```
define_memory_operation -module mem1pcell -memory_type 1p -write
-clock_and_polarity {CLK pos} -data_in D -address WADR -enable "ME"
-when "WE" -write_mask "4’b1010"

define_memory_operation -module mem1pcell -memory_type 1p -read
-clock_and_polarity {CLK pos} -data_out Q -address RADR -enable "ME"
-when "~WE"

define_memory_operation -module mem1pcell -sleep  -sleep_mode_when LS
```

Example 3: Defines memory interface information for different operations of the 1r1w memory module `mem1r1wcell`.

```
define_memory_operation  -module mem1r1w -memory_type 1r1w -read
-clock_and_polarity {RCLK pos} -data_out Q -address RADR -when "RE"

define_memory_operation  -module mem1r1w -memory_type 1r1w -write
-clock_and_polarity {WCLK pos} -data_in D -address WADR -when "WE"
-write_mask "4{1'b1}"

define_memory_operation -module mem1r1w -sleep -sleep_mode_when  LS
```

Example 4: Defines memory interface information for different operations of the 1r1w memory module `mem1r1w`. This example illustrates how multiple input and output ports can be specified along with their addresses.

```
define_memory_operation
    -module mem1r1w -memory_type 1r1w -read
    -clock_and_polarity {CLK pos}
    -data_out "Q15,Q14,Q13,Q12,Q11,Q10,Q9,Q8,Q7,Q6,Q5,Q4,Q3,Q2,Q1,Q0}"
     -address "{{3{1'b0}},RADR6,RADR5,RADR4,RADR3,RADR2,RADR1,RADR0}
              << 3}"
     -when "RE1 & RE2 & RE3"

define_memory_operation
    -module mem1r1w -memory_type 1r1w -write
    -clock_and_polarity {CLK pos}
    -data_in "{D15,D14,D13,D12,D11,D10,D9,D8,D7,D6,D5,D4,D3,D2,D1,D0}"
    -address "{{3{1'b0}},WADR6,WADR5,WADR4,WADR3,WADR2,WADR1,WADR0}
             << 3}"
    -when "WE1 & WE2 WE3"
    -write_mask "{~WM15,~WM14,~WM13,~WM12,~WM11,~WM10,~WM9,~WM8,~WM7,
        ~WM6,~WM5,~WM4, ~WM3,~WM2,~WM1,~WM0}"

define_memory_operation -module mem1r1w -sleep -sleep_mode_when "LS"
```

disable_msg

disable_msg — Disables reporting of a specific message ID. Note that the message will be disabled only if the message ID corresponds to an informational or warning message.

Synopsis

`disable_msg [ -after` *`<after>`* `]` *`<message_id>`*

-after *`<after>`*

Number of times to display a message before disabling further reporting of the message. The default is 0, which will immediately disable further reporting.

`<message_id>`

Message ID.

Related Commands

enable_msg, report_messages

Usage

The `disable_msg` command is used to turn off reporting of specific message types.

The `disable_msg` command immediately disables the reporting of a message unless the `after` option is specified. The `after` option allows the message to be disabled after it has been reported a specific number of times.

Examples

Example 1: Disable messages with the message ID, `CDBT-CGI`.

```
disable_msg CDBT-CGI
```

Example 2: Disable reporting of the message `CDBT-CGI` after it has been reported 17 times.

```
disable_msg  -after 17 CDBT-CGI
```

dont_use_high_fanout_signals

dont_use_high_fanout_signals — Marks the dont-use constraint on signals whose fanout value is greater than or equal to the threshold specified. Note that this command can only be called after `prototype_design`.

Synopsis

```
dont_use_high_fanout_signals -threshold <value>
    [ -exclude_high_fanout_signals <signal_list>]
```

-threshold *<value>*

> Specifies the maximum threshold value. The dont-use constraint is applied to signals whose fanout value is greater than or equal to the threshold value specified.

-exclude_high_fanout_signals *<signal_list>* ...

> A tcl list of signals to be excluded from the scope of this command. The dont-use constraint is not applied to these signals.

Related Commands

set_dont_use

Usage

The **dont_use_high_fanout_signals** command marks the dont-use constraint on signals that have a threshold value greater than equal to the value specified.

Examples

Example 1: Marks all signals that have a threshold value greater than or equal to `1` with the dont-use constraint.

```
dont_use_high_fanout_signals -threshold 1
```

Example 2: Marks all the signals that have a threshold value greater than or equal to 1 with the dont-use constraint. The signal `shift` is excluded from the list of signals considered by the command.

```
dont_use_high_fanout_signals -threshold 1    \
  -exclude_high_fanout_signals "shift"
```

dont_use_max_logic_depth_signals

dont_use_max_logic_depth_signals — Applies the dont-use constraint on signals whose depth is greater than or equal to the threshold value specified. Note that this command can only be called after `prototype_design`.

Synopsis

```
dont_use_max_logic_depth_signals -threshold <value>
    [ -exclude_high_depth_signals <signal_list>]
```

-threshold *<value>*

Specifies the threshold value to be used for comparison. The dont-use constraint is applied to the signals with a depth greater than or equal to the threshold specified.

-exclude_high_depth_signals *<signal_list>* ...

Specifies a Tcl list of signals to be excluded.

Related Commands

set_dont_use

Usage

The `dont_use_max_logic_depth_signals` command marks the dont-use constraint on signals that have a depth greater than or equal to the threshold specified.

Examples

Example 1: Marks signals that have a depth greater than or equal to `1` with the dont-use constraint.

```
dont_use_max_logic_depth_signals  -threshold 1
```

Example 2: Marks signals that have a depth greater than or equal to `1` with the dont-use constraint. The signal `shift` is excluded from the list of signals considered.

```
dont_use_max_logic_depth_signals  -threshold 1       \
      -exclude_high_depth_signals "shift"
```

enable_msg

enable_msg — Enables reporting of messages.

Synopsis

```
enable_msg [ <message_id> ]
```

<message_id>

Specifies the message ID for which reporting is to be enabled.

Related Commands

disable_msg, report_messages

Usage

The **enable_msg** command enables reporting for a disabled Message ID. If specified without an argument, reporting for all messages is enabled.

Examples

Example 1: Enables reporting of messages with the ID `APP-WDR`.

```
enable_msg APP-WDR
```

Example 2: Enables reporting for all messages.

```
enable_msg
```

exclude_flop_clock_gating

exclude_flop_clock_gating — Creates a script that lists flops that do not improve clock gating efficiency. The script can then be used as an input for DC and RC during clock gating. Clock gating such flops can lead to an increase in power.

Synopsis

```
exclude_flop_clock_gating [ -lp_tool<dc|rc>] [ -filename <file_name>]
    [ -power_diff <limit> |  -custom_score <limit> |  -efficiency
    <limit>] [ -bus_naming_style<string>] [
    -bus_range_separator_style<string>]
    [ -bus_dimension_separator_style<string>]
```

-lp_tool *<dc|rc>*

Specifies the tool for which the script is to be generated. Legal values are `dc` and `rc`. Default is `dc`.

-filename *<file_name>*

Specifies the name of the file to which the script is to be written. If not specified, the script is written to the standard output.

-power_diff *<limit>* | -custom_score *<limit>* | -efficiency *<limit>*

Specifies the criteria to be applied for identifying the flops to be excluded from clock gating. List of flops are written to a clock gating exclusion script by the file as specified by `-filename`.

Specify `-power_diff` to identify a list of flops, which when included, lead to an increase in power. The threshold is defined by *<limit>*. Value must be in `nW`.

Specify `-custom_score` to identify a list of flops whose custom score is less than the *<limit>* specified. Custom score is calculated as follows:

```
Custom Score = (width of flop) * (efficiency of that flop)
```

Specify `-efficiency` to identify a list of flops whose clock gating efficiency is less than the *<limit>* specified.

-bus_naming_style *<string>*

Specifies the naming convention for the arrays. Default is `%s[%d]` where, `%s` is used to substitute the name of the array while `%d` substitutes the index.

-bus_range_separator_style *<string>*

Specifies the array range separator. For example, in the range definition `a[5:7]`, the colon(:) is used as a separator. Default is `:`.

-bus_dimension_separator_style *<string>*

Specifies the separator for multi-dimensional arrays. For example, for the multi-dimensional array `a[5][3]`, **][** is used as a dimension separator. Default is `][`.

Related Commands

report_efficiency

Usage

The **exclude_flop_clock_gating** command is used to identify flops that do not improve clock gating efficiency. The list of flops are written to a script, which can then be used in DC/RC, to exclude less efficient flops from clock gating. Clock gating such flops can lead to an increase in power.

Examples

Example 1: Generates a script for use with RC.

```
exclude_flop_clock_gating  -lp_tool rc
```

A sample script generated for use with RC is shown below. As the `-filename` option has not been specified, results are written to the standard output.

```
###################### RC Script #########################
##bus_naming_style : %s[%d]
##suffix : _reg

proc exclude_cg {} {
    set flop [ find  -inst */count_inst/count_reg* ]
    if { $flop == ""t } {
        puts "count_inst/count_reg does not exist"
    } else {
        foreach reg $flop {
            set_attribute lp_clock_gating_exclude true $reg
        }
    }
}
```

Example 2: Generates a script for use with DC. Script is written to a file named `dc.tcl`.

```
exclude_flop_clock_gating  -lp_tool dc  -filename dc.tcl
```

A sample resultant script is shown below.

```
####################### DC Script ##########################
##bus_naming_style : %s[%d]
##bus_range_separator_style : :
##bus_dimension_separator_style : ][
##suffix : _reg

proc exclude_cg {} {
    set exclude_regs_list ""
    set flop [filter_collection [all_registers]
"full_name=~*count_inst/count_reg*" ]
    if { $flop == "" } {
        puts "count_inst/count_reg does not exist"
    } else {
        add_to_collection  $exclude_regs_list $flop
    }
    set_clock_gating_registers  -exclude $exclude_regs_list
}
```

exit

exit — Terminates PowerPro and returns control to the operating system shell.

Synopsis

exit [*<exit_code>*]

<exit_code>

Specifies the value returned by PowerPro on exit. If not specified, PowerPro returns an exit code of `0`. The exit code can be viewed using the UNIX command `echo $?`.

Usage

The **exit** command causes PowerPro to terminate and return control to the operating system shell. The exit code can be viewed using the UNIX command `echo $?`.

Examples

Example 1: Terminates PowerPro and returns an exit code of `1`.

```
powerpro> exit 1
0 warning, 0 error message
0.500u 0.171s 0.671 0:04.000   22.509m 41.375p
(PowerPro process used 41 MB and 1 seconds)

[/home/calypto]$ echo $?
1
[/home/calypto]$
```

Example 2: Terminates PowerPro and returns the default exit code, `0`.

```
powerpro> exit
0 warning, 0 error message
0.500u 0.171s 0.671 0:04.000   22.509m 41.375p
(PowerPro process used 41 MB and 1 seconds)

[/home/calypto]$ echo $?
0
[/home/calypto]$
```

filter_collection

filter_collection — This command filters an existing collection based on the specified filter, and results in a new collection. Note that the base collection remains unchanged. This command can only be run after `build_design`.

Synopsis

`filter_collection` *`<base_collection>`* *`<filter_expression>`*

`base_collection`

Specifies the base collection that must be filtered.

`filter_expression`

Specifies the expression that must be used to filter the base collection.

Related Commands

find, get_ports

Usage

The **filter_collection** command is used when there is a list of objects that must be filtered based on a specific object attribute.

Example

Example 1: Filters the collection of ports that matches the pattern abc* and are input type ports.

```
powerpro> filter_collection [get_ports abc*] {port_direction==in}
```

find

find — Queries the design database for information about modules, ports, signals and instances in the designs.

Synopsis

```
find {  -module |  -port |  -signal |  -inst }
    [ -graybox |  -blackbox |  -sequential |  -memory ]
    [ -input |  -output ] [ -user] [ -hier]
    [ -short_name |  -count ]
    [ -wildcard |  -exact |  -regexp ]
    [ -clock_domain<domain name> ]
    [ -clock_polarity< pos | neg > ] [ <expr> ]
```

-module | -inst | -port | -signal

Specifies the scope of search (modules, instances, ports or signals).

-blackbox

When searching for instances, restricts search to blackboxed instances.

-graybox

Reports all hierarchical instances which have been marked as grayboxes using the `set_gray_box` command.

-sequential

When searching for instances, restricts search to sequential elements i.e., flops, latches, and memories.

-memory

When searching for instances, restrict search to memories.

-input | -output

When searching for ports, restricts search to primary or blackbox input and output ports.

-user

Restricts search to objects present in the original design. Any additional objects generated by PowerPro during design interpretation and optimization are not included in the search.

-hier

Normally, the search pattern is applied at the top of the instance hierarchy. If `-hier` is specified, apply the search pattern to every hierarchical instance as well. Has no effect if searching for modules.

-short_name | -count

Type of information to be returned, either object names or a count (`-count`) of the number of objects found. Names may be returned as the original hierarchical path names (the default) or as short names (`-short_name`) with no hierarchical path information.

-wildcard | -exact | -regexp

Interpret the search expression using wildcard, exact, or regular expression matching rules. Wildcarding is the default matching style. Wildcarding extends UNIX shell `*` and `?` glob matching with a `**` operator which matches zero or more levels of hierarchy.

-clock_domain *<domain_name>*

This option will filter the results to those sequential instances which are part of specified clock -domain. Can only be used if " -inst" and " -sequential" options were also specified. This option can be used only after **prototype_design** command was executed successfully.

-clock_polarity *< pos | neg >*

This option will filter the results to those sequential instances with matching clock polarity. Can only be used if " -clock_domain" option was specified.

<expr>

A search expression which names a set of objects in the designs. The default search expression is "*". Bracket the expression, `{<expr>}`, when part -selecting or using escaped names which include `[`, `]`, or \ characters.

Related Commands

find_nodes

Usage

The `find` command searches the database to find objects that match the given search criteria. Other commands that accept hierarchical names to access design objects use this command internally, so all the naming rules which apply to this command also apply to those commands.

There are four types of information required for each search:

- Kind of object to search
- Scope of the search
- Output format
- Search expression

Kind of Object

The `find` command searches the following objects:

- module
- inst
- port
- signal

Note that at least one must be specified.

Instance searches may be restricted to sequential (`-sequential`) or blackboxed (`-blackbox`) objects. Sequential objects are explicitly specified or implicitly inferred flops, latches, and memory elements present in a design. Black boxes are all module instances blackboxed by the user.

Using the `-user` option, all searches may be restricted to objects present in the original design. Any objects generated during module optimizations are not included.

Scope of the search

The scope of the search must be restricted to the design.

Normally, the search pattern is applied from the top of the instance hierarchy. If `-hier` is specified, the search pattern is applied from every hierarchical instance in the design. For example:

```
find  -port port_A
```

will only find a `port_A` port at the top instance of the design, while:

```
find   -hier  -port port_A
```

will find `port_A` ports of all instances in the design.

Output format

A search result is returned as a list or count of objects found. If no object matches, an empty list or zero count is output. There are 3 output formats:

default	original hierarchical names
-short_name	names without hierarchical information
-count	count of objects found

The default is by list of original hierarchical names.

Search expression

A search expression matches the object to be described. The search expression may be interpreted using wildcard, exact, or regular expression semantics. The default is wildcard semantics.

See the Appendix for more details on wildcard and regexp matching rules.

The search expression for `-instance`, `-port` and `-signal` is the hierarchical name of that object. The search provides objects in a uniquified/flattened view of the module. For example,

```
find  -port  i5.i2.*
```

returns all ports of instance `i5.i2` in the design:

```
i5.i2.p0 i5.i2.p1
```

Note that `*` wildcarding will not cross levels of hierarchy. In the previous example, ports of instances below `i5.i2` will not be found. Instead, use `**` wildcarding:

```
find  -port  i5.**.p*
```

returns all ports beginning with `p` which are in instance `i5` or any instances contained in `i5`:

```
i5.p3 i5.i1.p0 i5.i2.p0 i5.i2.p1 i5.i2.i3.p0  i5.i2.i3.p2
```

The search expression for a module is the name of the module only. For example:

```
find   -module mod_a*
```

This returns all the modules in the design with `mod_a` as the prefix:

```
mod_adder mod_addmult mod_alu
```

Some object types, such as memories, may be flattened by PowerPro. Searches can use the original names, but the flattened object names will be returned.

For example, memory arrays are blasted into registers in the database. The registers will be assigned blasted names. The following example shows all the kinds of searches that can be performed. Assume the following memory is instantiated in the design:

```
reg [7:0]          mem[3:0];                // Verilog
```

then:

```
powerpro> find   -inst mem
      mem[0], mem[1], mem[2], mem[3]
powerpro> find  -inst mem\[2\]
      mem[2]
```

Because PowerPro tries to work at the word level, if a design depends on an object, but not on any specific bit or part of that object then PowerPro may not flatten the object into its bit level representation. In that case, only the object itself may be searched for; a specific bit or part select may not be searched for.

Signal Aliasing

When PowerPro builds a design database, some limited optimizations are done on logic inside design modules. As a result of these optimizations, some internal signals may be optimized away, while other internal signals may be merged because they have the same functionality. Ports will not be affected.

When a signal is optimized away, it will never appear in a search result list.

If two or more signals become merged, one of the original signal names is chosen to be the name of the merged signal. Other signal names become aliased to that name. Searching for any one of the original signal names will always return the name of the merged signal.

find_lib_cell

find_lib_cell — Returns the names of library cells matching the expression specified. Note that this command can only be called after `read_library`.

Synopsis

`find_lib_cell` -cell *<cell_name>* [-exact | -wildcard]

-cell *<cell name>*

Specifies the name of the library cell to find. Supports wildcards.

-exact | -wildcard

Specifies the type of match i.e., wildcard match or exact match. Default is `wildcard`. Note that wildcarding extends UNIX shell behavior of * and ? glob matching with a ** operator which matches zero or more levels of hierarchy.

Related Commands

set_threshold_voltage_group

Usage

The `find_lib_cell` command returns the names of the library cells matching the argument.

Examples

Example 1: Assuming a design contains the following cells: lib1/AND1, lib1/AND2, lib1/AND2x, lib2/AND1, lib2/AND2 and lib2/AND2x, this command will return the following cells: lib1/AND1 lib1/AND2 lib1/AND2x.

```
find_lib_cell  -cell lib1/AND*
```

As the cell name is specified with a wildcard, this command is equivalent to specifying:

```
find_lib_cell  -cell lib1/AND*  -wildcard
```

find_nodes

find_nodes — Returns the flops found in the fanin or fanout cone of a given list of signals. Note that this command can only be called after `build_design`.

Synopsis

```
find_nodes { -fanin | -fanout } { -start_points<start_point_list> }
    { -depth <depth> | -end_points <end_point_list> } [ -hier ]
```

-fanin | -fanout

Specifies whether the flops are searched in the fanin or fanout cone.

-start_points*<start_point_list>*

Specifies the list of starting signals. The fanin or fanout cones of all the signals specified in the starting signal list are returned.

-depth *<depth>*

Specifies the sequential depth for the fanin/fanout traversal. A depth of 1 implies that the traversal should stop at the first flop boundary. This option cannot be used in conjunction with `-end_points`.

-end_points *<end_point_list>*

Specifies the list of end signals. While tracing the fanin or fanout cone, the search for a flop is made until an end point is reached or a primary input, primary output or a black box port is reached. This option cannot be used in conjunction with –`depth`.

-hier

By default, the search pattern is applied at the top of the instance hierarchy. If -hier is specified, the search pattern is applied to every hierarchical instance as well.

Related Commands

find

Usage

The find_nodes command searches for flops in the fanin or fanout cone of a given list of signals and returns the list of flops found. The result is the union of all the flops, found by tracing the fanin or fanout cone of every start point specified in the start signal list until one of the end points, a primary input/output port, a black box port or the specified sequential depth is reached. If a flop is in the fanin or fanout cone of multiple start points then it is listed only once. Consider the following design example.

```
module top(out, clk, d, reset);
output out;
input clk;
input d;
input reset;
reg fl;
always @(posedge clk)
    begin
    if (reset == 1'b1)
        fl <= 0;
    else
        fl <= d;
    end
    assign out  = !fl;
endmodule
```

Using the above design, the following command reports all flops in the fanin cone of signal out until a sequential depth of 1.

```
find_nodes -fanin -start_points "out" -depth 1
Returns: fl
```

Likewise, the following command returns all flops which are in the fanout cone of d and also in the fanin cone of out.

```
find_nodes -fanout -start_points "d" -end_points "out"
Returns: fl
```

Examples

Example 1: Reports all flops present in the fanout of ports spec.in and spec.t1.in until the logical depth 5. Hierarchical instance, if any, present in fanout of ports spec.in and spec.t1.in will be skipped.

```
find_nodes -start_points {spec.in spec.t1.in} -fanout -depth 5
```

Example 2: Reports all flops present in the fanin of port `spec.out` until the end points `spec.in` or `spec.t1.in` or a primary input or a black box port is reached. Additionally, because the -hier option has been used, all hierarchical instances present in fanin of port `spec.out` will also be considered.

```
find_nodes -start_points {spec.out}
    -end_points {spec.in spec.t1.in} -fanin   -hier
```

generate_guidance

generate_guidance — Generates guidance and early design check reports in PowerPro. The command can be run only after the `prototype_design` command and cannot be run after any optimization command is run.

Synopsis

```
generate_guidance [-early_design_checks] [-combinational ] [-uarch ] [-
    all ]
```

-early_design_checks

This option controls the generation of early design check reports. If this option is specified, the command generates early design check reports.

-combinational

This option controls the generation of combinational reports. If this option is specified, the command generates combinational reports.

-uarch

This option controls the generation of micro-architectural reports. If this option is specified, the command generates micro-architectural reports.

-all

If this option is specified, the command generates guidance (combinational and micro-architectural) and early design check reports. If no option is specified with the command, this will become the default option.

Related Commands

config_guidance_reports, report_html_format, show_analyzer

Usage

Generates guidance and early design check reports in PowerPro. If no option is specified, all the guidance and early design check reports will be generated. If the command is run after the `prototype_design` command and before the `report_power` command, only early design check reports will be generated.

Example

Example 1: Generates the early design check reports.

```
powerpro> generate_guidance -early_design_checks
[PPRO-GARI]  Generating Early Design Check Reports.
```

---

get_cells

get_cells — Returns a collection of cells from the current design. Note that this command can only be run after the `build_design` command.

Synopsis

`get_cells` [-hierarchical] [-regexp] [<*pattern*>]

-hierarchical

When this option is specified, the search applies to all the hierarchical instances as well. By default, the search is applied at the top of the instance hierarchy.

-regexp

When this option is specified, a regular expression must be used to perform the search. Default is plain search.

<*pattern*>

Specifies the pattern that must be used for matching the cell names. Pattern matching is performed using the wilcard matching rules. Note that the search is case-sensitive and the wildcard character `*` is supported.

Note: In case a pattern is not specified, `get_cells` returns all cells in the design.

Related Commands

find, get_lib_cells

Usage

The **get_cells** command returns a list of cells that matches the specified pattern. In case no cells match the specified pattern, an empty list is returned.

Examples

Example 1: Prints the cells that exist in all hierarchies.

```
get_cells -hierarchical
```

Example 2: Prints the cells that match the pattern `*p1*` in all hierarchies.

```
get_cells -hierarchical *p1*
```

Example 3: Prints the cells that match the regular expression `.*p1.*`.

```
get_cells -regexp .*p1.*
```

get_clocks

get_clocks — Returns a list of clocks in the design, matching the specified pattern. Note that this command can only be called after `prototype_design`.

Synopsis

`get_clocks [<pattern>]`

`<pattern>`

Specifies the pattern to be matched. Pattern matching is performed using wilcard matching rules. Note that the search is case-sensitive and that the wildcard character `*` is supported.

Note: In case a pattern is not specified, `get_clocks` returns all clocks in the design.

Related Commands

all_clocks, create_clock, create_generated_clock, find

Usage

Returns a list of clocks that have been created using the `create_clock` or `create_generated_clock` command. A pattern can be specified to restrict the list of clocks returned. Wildcard matching rules apply.

The list returned includes ports which are treated as clock -roots in the design. In case, no clock -roots match the pattern specified, an empty string is returned.

Examples

Example 1: Returns a list of all clocks in the design.

```
get_clocks *
```

get_clock_gating_info

get_clock_gating_info — Returns clock gating statistics for sequential elements in the design. Note that this command can be called only after the `prototype_design` command.

Synopsis

```
get_clock_gating_info [ -inst <instance_name> ] [ -efficiency |
      -observability |  -stability |  -sequential]
    [ -enable |  -optimize |  -patched |  -stable_type ]
    [-cg_min_size_based]
```

-inst *<instance_name>*

Restricts the scope of the command to the sequential instance specified by `instance_name`. When specified with the `-enable`, `-optimize`, `-patched`, or `-stable_type` option, *<instance_name>* must be the full hierarchical name of the flop.

-efficiency | -observability | -stability | -sequential

Specify `-efficiency` to return the clock -gating efficiency of the sequential element specified.

Specify `-observability` to return the number of sequential elements gated during observability based sequential optimization.

Specify `-stability` to return the number of sequential elements gated during stability based sequential optimization.

Specify `-sequential` to return the number of sequential elements gated during observability and stability based sequential optimization. The value returned is a sum of the values returned by the `-observability` and `-stability` options.

-enable | -optimize | -patched | -stable_type

Note: These options expect the hierarchical name of the flop to be specified using the `-inst` option.

Note: PPRO-24744

Specify `-enable` to report whether the write enable logic of the flop is user specified, new, strengthened or always on. The values returned are `user`, `new`, `strengthened` and `always -on`.

Specify `-optimize` to report whether or not a dont-optimize constraint had been set on a flop. Values returned are `yes` and `no`.

Specify `-patched` to report whether or not the optimized flop was patched into the `write_rtl` output. Values returned are `yes` and `no`.

Specify `-stable_type` to report the type of stability-based enable logic applied on a flop. Returns `constant` if a constant stability-based enable was applied, `symbolic` if a symbolic stability-based enable was applied, and `n/a` if no stability-based enable was applied.

-cg_min_size_based

Specify `-cg_min_size_based` to return the clock gating efficiency of the sequential instance based on the global `opt_cg_min_size`. This option impacts only the efficiency of the sequential element.

Related Commands

report_efficiency

Usage

The **get_clock_gating_info** command returns clock -gating related statistics for sequential elements in the design. If no options are specified, a Tcl list of statistics is returned. Otherwise, a single value is returned.

Examples

Example 1: Returns clock gating statistics for sequential elements in the design.

```
powerpro> get_clock_gating_info
{efficiency 94.56 seq 0 obs 0 stb 0 comb 0}
```

Example 2: Returns the number of sequential elements gated during observability based sequential clock gating.

```
powerpro> get_clock_gating_info  -observability
21
```

Example 3: Returns the clock gating efficiency of the register bank `bank1`.

```
powerpro> get_clock_gating_info  -efficiency  -inst bank1
78.9
```

Example 4: Returns the type of write enable logic for the flop `p11.w1.d`.

```
powerpro> get_clock_gating_info  -enable  -inst p11.w1.d
new
```

Example 5: Returns clock gating statistics of the sequential elements based on the `cg_min_size_based` option.

```
powerpro> get_clock_gating_info -cg_min_size_based
{efficiency 60 seq 0 obs 0 stb 0 comb 0}
```

Example 6: Returns the clock gating efficiency of the sequential element `p11.w1` based on the `cg_min_size_based` option.

```
powerpro> get_clock_gating_info -inst p11.w1 -efficiency -
cg_min_size_based
80
```

Example 7: Returns the clock gating efficiency of the sequential elements based on the `cg_min_size_based` option.

```
powerpro> get_clock_gating_info -efficiency -cg_min_size_based
60
```

get_frequency

get_frequency — Returns the frequency (in `MHz`) for the specified signal or port. The command must be called after the `prototype_design` command.

Synopsis

`get_frequency {-signal <signal_name> | -port <port_name>} [-average]`

`-signal <signal_name> | -port <port_name>`

Specifies the hierarchical name of the signal or port for which the frequency must be returned.

`-average`

Returns the average frequency of the specified multi-bit signal or port.

Note: By default, the `get_frequency` command returns the bit-wise frequency.

Related Commands

read_fsdb, read_saif, get_sa

Usage

The `get_frequency` command returns the frequency (in MHz) for the specified signal or port.

Example

Example 1: Returns the frequency of the port `in1`.

```
build_design input.v
read_saif input.v.saif
prototype_design
get_frequency -port in1

Result:
powerpro> get_frequency -port in1
460.153
```

Example 2: Returns the frequency of the signal `sig1`. Assuming that the signal `sig1` is a 2-bit signal, two values are returned.

```
build_design input.v
read_fsdb input.v.fsdb
prototype_design
get_frequency -signal sig1

Result:
powerpro> get_frequency -signal sig1
458.955  448.483
```

Example 3: Returns the average frequency of the 2-bit signal `sig1`.

```
build_design input.v
read_saif input.v.saif
get_frequency -signal sig1 -average

Result:
powerpro> get_frequency -signal sig1 -average
453.719
```

---

get_global

get_global — Returns a global variable value.

Synopsis

`get_global <name>`

`<name>`

Name of the global variable whose value is required.

Related Commands

set_global

Usage

Globals allow the user to alter the default behavior of PowerPro. Use the **get_global** command to return the current value of a global. For a complete list of globals supported and their descriptions, refer to the PowerPro Reference Manual.

Example

Example 1: Returns the value of the global `opt_cg_min_size`. Assuming the `set_global` command was specified as follows:

```
powerpro> set_global opt_cg_min_size 1
1
```

Specifying the `get_global` command will return `1`.

```
powerpro> get_global opt_cg_min_size
1
```

get_lib_cells

get_lib_cells — Returns a list of all cells present in read libraries.

Synopsis

```
get_lib_cells
```

Related Commands

get_libs, find

Usage

The **get_lib_cells** command returns a list of all cells present in read libraries.

Example

Example 1: Returns a list of all cells present in the library.

```
powerpro> get_lib_cells
```

get_libs

get_libs — Returns a list of all read libraries.

Synopsis

```
get_libs
```

Related Commands

get_lib_cells, find

Usage

The **get_libs** command returns a list of all read libraries.

Example

Example 1: Returns a list of all read libraries.

```
powerpro> get_libs
```

get_load

get_load — Returns capacitance information for a signal or port. Note that this command can be called only after the `prototype_design` command.

Synopsis

```
get_load {-signal <signal_name> | -port <port_name>}
```

-signal <*signal_name*> | -port <*port_name*>

> Specifies the hierarchical name of the signal or port for which capacitance information must be obtained.

Related Commands

read_spef, report_power, set_load

Usage

The **get_load** command can be used to print the capacitance information for signals and ports.

Examples

Example 1: Returns the capacitance for the port `in1` in the design.

```
powerpro> get_load -port in1
0.0460153
```

Example 2: Returns the capacitance for the user signal `sig1` in the design. Assuming that the signal `sig1` is a 2-bit signal, two values are returned.

```
powerpro> get_load -signal sig1
0.458955  0.448483
```

Example 3: Returns the capacitance for the output port `q` of the flop `hier1.hier2.ff1` in the design.

```
powerpro> get_load -port hier1.hier2.ff1.q
0.45432
```

Example 4: Returns the capacitance for the output port `Q` of the tech-flop `hier1.hier2.TECH_FLOP_INST` in the design.

```
powerpro> get_load -port hier1.hier2.TECH_FLOP_INST.Q
0.25432
```

get_mem_gating_info

get_mem_gating_info — Returns memory -gating statistics for memories in the design. Note that this command can only be called after `prototype_design`.

Synopsis

```
get_mem_gating_info [ -inst <instance_name> ] [ -observability |
     -stability |  -sleep |  -sequential |  -efficiency ]
```

-inst *`<instance_name>`*

Restricts the scope of the command to the design instance specified by `instance_name`. If *`<instance_name>`* corresponds to a memory in the design, information for that memory is printed.

-observability | -stability | -sleep | -sequential | -efficiency

Specify `-observability` to return the number of memories gated during observability based sequential optimization.

Specify `-stability` to return the number of memories gated during stability based sequential optimization.

Specify `-sleep` to return the number of memories gated during light sleep based sequential optimization.

Specify `-sequential` to return the number of memories that are gated during observability or stability based sequential optimization. The results are equivalent to adding up the values returned by the `-observability` and `-stability` options.

Specify `-efficiency` to return the memory -gating efficiency of the memory.

Related Commands

report_memory_efficiency

Usage

The **get_mem_gating_info** command returns memory gating statistics for memories in the design. If no options are specified then a Tcl list of statistics is returned. Otherwise, a single value is returned.

Examples

Example 1: Returns memory gating statistics for the design.

```
powerpro> get_mem_gating_info
{rd_efficiency 60.42 sleep_efficiency 50 seq 6 obs 3 stb 3 ls 0}
```

Example 2: Returns the number of memories gated during observability based memory -gating.

```
powerpro> get_mem_gating_info  -observability
3
```

Example 3: Returns the Memory Gating Efficiency and Memory Sleep Mode Efficiency of the memory `mem_inst1`.

```
powerpro> get_mem_gating_info  -efficiency  -inst mem_inst1
78.0 42.0
```

get_nets

get_nets — Returns a list of nets matching the pattern specified. Note that this command can only be called after `build_design`.

Synopsis

```
get_nets [ -hierarchical] [<pattern>]
```

-hierarchical

By default, the search is applied at the top of the instance hierarchy. When this option is specified, the search applies to hierarchical instances as well.

<pattern>

Specifies the pattern to use for matching net names. Pattern matching is performed using wilcard matching rules. Note that the search is case-sensitive and that the wildcard character `*` is supported.

Note: In case a pattern is not specified, `get_nets` returns all nets in the design.

Related Commands

find, get_ports

Usage

The **get_nets** command returns a list of nets matching the pattern specified. The list of nets returned depends on the current instance. In case no nets match the pattern specified, an empty list is returned.

Examples

Example 1: Marks a list of nets matching the name `inst1.blockx1*` with the dont-use constraint.

```
set_dont_use [ get_nets inst1.blockx1* ]
```

get_ports

get_ports — Returns a list of ports matching the pattern specified. Note that this command can only be called after `build_design`.

Synopsis

```
get_ports [ -hierarchical] [<pattern>][-filter expression]
```

-hierarchical

By default, the search is applied at the top of the instance hierarchy. When this option is specified, the search applies to hierarchical instances as well.

<pattern>

Specifies the pattern to use for matching port names. By default, the pattern

matching is performed using wildcard matching rules. Note that the search is case-sensitive and that the wildcard character `*` is supported.

Note: In case a pattern is not specified, `get_ports` returns all ports in the design.

-filter expression

Filters the list of ports based on the specified expression.

Related Commands

find, get_nets

Usage

The **get_ports** command returns a list of ports matching the pattern specified. The list of ports returned depends on the current instance. In case no port matches the pattern specified, an empty list is returned.

Examples

Example 1: Prints switching activity information for inputs ports beginning with `in1`.

```
foreach port [get_ports in1*] {
    set td_val [get_sa  -port $port  -td]
    set prob_val [get_sa  -port $port  -prob]
    puts "$port : td=$td_val  prob=$prob_val"
}
```

Example 2: Prints the input ports of the current module.

```
get_ports -filter port_direction==in
```

Example 3: Prints the output ports of the current module.

```
get_ports -filter port_direction==out
```

get_sa

get_sa — Extracts switching activity information for a signal or port. For signals or ports where switching activity data has been annotated using the `read_saif`, `read_fsdb`, or `set_sa` command, `get_sa` can be called any time after `build_design`. For all other signals or ports, `get_sa` must be called after `prototype_design`.

Synopsis

```
get_sa { -signal <signal_name> |  -port <port_name> }
    { -td |  -prob } [ -source]
```

-signal *<signal_name>* | -port *<port_name>*

Specifies the hierarchical name of the signal or port for which switching activity information is to be obtained.

-td | -prob

Specifies the type of switching activity required.

Specify `-td` to extract toggle density values annotated on the specified port/signal.

Specify `-prob` to extract probability values annotated on the specified port/signal.

-source

Reports source relation i.e., a list of `<saif_file,line_number>`, from which the switching activity was annotated during `read_saif` for the specified port or signal.

Note: When this option is specified with the `-td` or the `-prob` option, the actual probability or toggle density value is not returned. Instead, the source file and line number from which the switching activity was annotated during `read_saif` is returned.

Related Commands

read_fsdb, read_saif, set_sa

Usage

The **get_sa** command extracts switching activity information for signals and ports.

Examples

Example 1: Returns the toggle density value annotated on the port `in1` from the SAIF file.

```
build_design input.v
read_saif input.v.saif
get_sa  -td  -port in1
```

Result

```
powerpro> get_sa  -td  -port in1
0.0460153
```

Example 2: Returns the probability value annotated on the `sig1` by PowerPro. Assuming the signal `sig1` is a 2 bit signal, 2 values are returned.

```
build_design input.v
prototype_design
get_sa  -prob  -signal sig1
```

Result

```
powerpro> get_sa  -prob  -signal sig1
0.458955  0.448483
```

Example 3: Returns the file and line number in the SAIF file using which the probability value was annotated on the signal `sig1`. In this case, the results show line 81 in the file `input.v.saif`.

```
build_design input.v
read_saif input.v.saif
get_sa  -prob  -signal sig1  -source
```

Result

```
powerpro> get_sa  -prob  -signal sig1  -source
{input.v.saif 81}
```

get_transition

get_transition — Returns slew (max) values for all the bits of the specified port. Note that this command can only be called after `prototype_design`.

Synopsis

```
get_transition { -rise | -fall } <port name>
```

-rise

Specifies if rise slew (max) is to be obtained.

-fall

Specifies if fall slew (max) is to be obtained.

<*port name*>

Specifies the name of the port.

Related Commands

set_transition

Usage

Gives as output the slew value of rise or fall of the port specified.

Examples

Example 1: Gives the rise slew value for the port abc.in.

```
get_transition -rise abc.in
```

Example 2: Gives the fall slew value for the port abc.in.

```
get_transition -fall abc.in
```

get_version

get_version — Returns the version number of the PowerPro release.

Synopsis

```
get_version [ -major |  -minor |  -patch ]
```

-major

Returns the major release number.

-minor

Returns the minor release number.

-patch

Returns the patch release number.

Usage

The **get_version** command returns the PowerPro version number or a specified field thereof.

Example

The version number of any PowerPro release is composed of multiple fields.

For example, the release `3.0 -prod -4` has a major number `3`, minor number `0` and the patch number is `4`.

---

insert_cg_marker_cell

insert_cg_marker_cell — Inserts marker cells within the clock gating hierarchies generated by PowerPro. New Marker cells are inserted just before the output ports of the clock gating hierarchies. Note that this command can only be called after all sequential optimization commands have been executed.

Synopsis

```
insert_cg_marker_cell [ -inst <inst_list> ] {  -cell <cell_name> }
```

-inst *<inst_list>*

Inserts marker cells inside all Clock Gating Hierarchies within the instance *<inst_list>*. If not specified, the default behavior applies - new marker cells are created for all the Clock Gating Hierarchies in the design.

-cell *<cell_name>*

Specifies the name of the cell that will be used to instantiate the marker cell. PowerPro looks for this cell in all modules of the constraints library (as specified by `build_design -cons`).

Related Commands

insert_mem_clock_gating, insert_mem_observability_gating, insert_mem_sleep_gating, insert_mem_stability_gating, insert_observability_logic, insert_stability_logic

Usage

Inserts marker cells within the clock gating hierarchies generated by PowerPro. New Marker cells are inserted just before the output ports of the clock gating hierarchies.

A strict check applies to the structure of the marker cell. Make sure the marker cell complies with the following guidelines.

- A marker cell can have only 1 input and 1 output port.
- A marker cell can have only single-bit input and output ports.
- A marker cell can have logic only for a feedthrough wire. No other logic is allowed.
- In case of VHDL designs, the input and output port of the marker cell must be `std_logic_vector(0 downto 0)`.

A sample RTL to define a marker cell in Verilog and VHDL is provided below.

Verilog RTL

```
module marker_cell ( input in, output out );
  assign out = in;
endmodule
```

VHDL RTL

```
LIBRARY IEEE ;
USE IEEE.std_logic_1164.ALL ;
USE IEEE.std_logic_unsigned.ALL ;

entity marker_cell is
    port( in1  : in std_logic_vector(0 downto 0);
          out1 : out std_logic_vector(0 downto 0)
        );
end marker_cell;

architecture arch_marker_cell of marker_cell is
begin
    out1 < in1;
end arch_marker_cell;
```

For the above mentioned RTL, the following command can be called.

```
insert_cg_marker_cell  -cell marker_cell
```

Consider another example. This example illustrates the complete flow as well as highlights what is not allowed.

```
build_design design.v

# Has to be specified using -cons option
build_design -cons marker.v

create_clock -period 1.33 -name clk clk
prototype_design
insert_observability_logic

# inserts marker cells in all Clock Gating Hierarchies
insert_cg_marker_cell -cell test_marker_cell

write_rtl

# This will cause an error
insert_stability_logic

write_verify_script
```

insert_mem_clock_gating

Deprecated w.e.f. PowerPro 10.4.

insert_mem_clock_gating — Clock-gates 1p and 1r1w power opt memories in the design. Note that this command can only be called after successfully executing the `prototype_design` command.

Synopsis

```
insert_mem_clock_gating
```

Related Commands

report_clock_gated_memories, set_cell_as_memory, set_module_as_cgic, set_scan_enable, set_target_cgic

Usage

The `insert_mem_clock_gating` command clock gates all `1p` and `1r1w` power opt memories in the design. To clock gate the memory with its corresponding enable signal and apply the gated clock to the memory, `insert_mem_clock_gating` introduces a user specified CGIC cell.

The `insert_mem_clock_gating` command allows you to achieve additional power savings on the memory by further gating the clock after all PowerPro optimizations have been done.

To enable PowerPro to clock gate the memory with the most optimized enable expression, no other sequential optimization command must be called after `insert_mem_clock_gating`.

Example

Example 1: Clock gates instances of the module `p1024x32m`.

```
set_module_as_cgic  -module my_CG_MOD  -clock ck_in
-clock_gate_enable_pin enable  -clock_gate_out_pin ck_out
-clock_gate_test_pin test

set_scan_enable -default -signal SCAN_EN

set_cell_as_memory  -module p1024x32m  -enable_polarity_clock "me pos
clk"  -clock_and_polarity "clk pos"

set_target_cgic -default my_CG_MOD

insert_mem_clock_gating
```

In this example,:

- The `insert_mem_clock_gating` command instantiates the CGIC cell `my_CG_MOD` in the CG hierarchy `cg_mem_cgic_`*`<parent_hier>`*, which will further be instantiated in the parent hierarchy *`<parent_hier>`* and connects the input pins `en` and `clk` to the corresponding memory enable `me` and `clk`. Further, it disconnects the input clock pins of the memory and connects them to the corresponding output clocks of the CGIC cell instances.
- The CGIC scan enable pin `test` is connected to the design scan enable `SCAN_EN`. All other pins of the CGIC cell instances are connected to `0`.

insert_mem_observability_gating

insert_mem_observability_gating — Performs observability-based sequential optimizations to shut off redundant reads to the memory in the current design. This results in dynamic power reduction for these memories. Note that this command can only be called after `prototype_design`.

Synopsis

```
insert_mem_observability_gating [ -effort low | medium | high ]
    [ -objective area | efficiency ] [ -max_area_limit <area_limit> ]
    [ -threshold <threshold_value> ] [ -logic_depth <ldepth> ] [ -
    max_support <size> ]
```

-effort low | medium | high

Specifies the relative amount of CPU time spent during observability-based sequential optimization. Legal values are `low`, `medium` and `high`. Default is `medium`.

-objective area | efficiency

Specifies the objective to be maximized during observability-based sequential optimization. Legal values are `area` and `efficiency`. Default is `-efficiency`.

Specify `area` as the objective to commit only those transformations that minimize the increase in area.

Specify `efficiency` as the objective to commit only those transformations that maximize the efficiency metric of the design.

-max_area_limit *<area_limit>*

Specifies the maximum acceptable increase in area, as a result of the enable logic added. *<area_limit>* is a float representing the fraction of the initial area of the design. Default is `0.5`, indicating that the increase in area as a result of the new enable logic will not exceed 50% of the original design area.

-threshold *<threshold_value>*

Specifies that only those transformations be committed whose score exceeds the *<threshold>* specified. *<threshold_value>* must be a valid float value between `0` and `1.0`, indicating the goodness of a transformation (as specified by `-objective`). Default is `0.0`.

-logic_depth <*ldepth*>

Ensures that the core enable logic depth does not exceed the value specified by the argument `ldepth`. Logic depth is defined as the depth of the logic when implemented using two-input gates. By default, logic depth is unconstrained.

-max_support <*size*>

Specifies the maximum number of bit-level inputs for the core enable logic. By default, size is unconstrained.

Related Commands

define_memory_operation, insert_mem_clock_gating, insert_mem_sleep_gating, insert_mem_stability_gating, report_memory_gating

Usage

The `insert_mem_observability_gating` command performs observability-based sequential optimizations to shut off loading of memories in the current design to reduce dynamic and static memory power consumption. Information about the enable logic added by PowerPro to shut off redundant loading of memories is written to the *<work*

`-`*`directory`*`>/enLogic.(v/vhd)` file. For more information about the `enlogic.(v/vhd)` file, refer to the PowerPro User Manual.

Invoking the `insert_mem_observability_gating` command without any options is equivalent to specifying:

```
insert_mem_observability_gating  -effort medium  -objective efficiency
      -max_area_limit 0.5  -threshold 0.0
```

This means that sequential optimizations will be performed with medium effort to maximize efficiency using only observability-based sequential transformations, where the additional area from the enable logic will not exceed 50% of the original design area, and all transformations whose goodness value exceeds 0.0 will be committed.

Examples

Example 1: Specifies that observability based clock gating be performed, with medium effort, for those transformations whose goodness value exceeds 0.2, at the same time ensuring that the increase in area as a result of the additional enable logic does not exceed 5% of the original design area.

```
insert_mem_observability_gating  -threshold 0.2  -max_area_limit 0.05
```

insert_mem_sleep_gating

insert_mem_sleep_gating — Performs Light Sleep based sequential optimizations to disable loading of memories in the current design to reduce static power memory. Note that this command can only be called after `prototype_design`.

Synopsis

```
insert_mem_sleep_gating [ -objective area | efficiency ]
    [ -max_area_limit <area_limit> ] [ -threshold <threshold> ]
```

-objective area | efficiency

Specifies the objective that is to be maximized by the sequential optimization. If the specified objective is `area`, the command commits those sequential transformations that minimize the area increase. If the objective is `efficiency`, the command commits those transformations that maximize the efficiency metric of the design. The default objective for light sleep based sequential optimization is `efficiency`.

-max_area_limit *`<area_limit>`*

Ensures that the area increased by the additional enable logic is smaller than the limit. The `area_limit` option is a float representing the fraction of the initial area of the design. The default is 0.5, meaning the area of the additional enable logic will not exceed 50% of the original design area.

-threshold *`<threshold>`*

Commits those transformations whose score exceed `threshold`. The score of a transformation is a float between 0 and 1.0 indicating the goodness of the transformation. It is based on the objective that the user provides. The `threshold` value is a float representing a valid score. The default is 0.0.

Related Commands

define_memory_operation, insert_mem_clock_gating, insert_mem_observability_gating, insert_mem_stability_gating, report_memory_gating

Usage

The `insert_mem_sleep_gating` command creates a `<`*`workdir`*`>/enLogic.(v/vhd)` file with information about enable logic added by PowerPro to shut off redundant reading of memories. For more information about the `enlogic.(v/vhd)` file, refer to the PowerPro User Manual.

Invoking the `insert_mem_sleep_gating` command with no options is equivalent to invoking it as:

```
insert_mem_sleep_gating  -effort medium  -objective efficiency
-max_area_limit 0.5  -threshold 0.0
```

In other words, sequential optimizations are performed with medium effort to maximize the efficiency metric using only light sleep based sequential transformations, where the additional area from the enable logic will not exceed 50% of the original design area, and all transformations whose goodness value exceeds 0.0 will be committed.

Examples

Example 1: Performs light sleep based sequential optimization for those transformations whose score exceeds 0.2 until the area consumed by the additional enable logic does not exceed 5% of the total design area.

```
insert_mem_sleep_gating  -threshold 0.2  -max_area_limit 0.05
```

insert_mem_stability_gating

insert_mem_stability_gating — Performs stability based sequential optimizations to shut off loading of memories in the current design to reduce dynamic power. Note that this command can only be called after `prototype_design`.

Synopsis

```
insert_mem_stability_gating [ -type<typeStr> ]
    [ -effort low | medium | high ] [ -objective area | efficiency ]
    [ -max_area_limit <area_limit> ] [ -threshold <threshold_value> ]
```

-type*`<typeStr>`*

Specifies the type of stability based sequential optimization to be performed. Legal values for *`<typeStr>`* are `C` (Constant Stability) or `S` (Symbolic Stability). Default is type `C` followed by type `S`.

-effort low | medium | high

Specifies the relative amount of CPU time spent during stability based sequential optimization. Legal values are `low`, `medium` and `high`. Default: `medium`.

-objective area | efficiency

Specifies the objective to be maximized during stability based sequential optimization. Legal values are `area` and `efficiency`. Default is `efficiency`.

Specify `area` as the objective to commit only those transformations that minimize the increase in area.

Specify `efficiency` as the objective to commit only those transformations that maximize the efficiency metric of the design.

-max_area_limit *`<area_limit>`*

Specifies the maximum acceptable increase in area, as a result of the enable logic added. *`<area_limit>`* is a float representing the fraction of the initial area of the design. Default is `0.5`, indicating that the increase in area as a result of the new enable logic should not exceed 50% of the original design area.

-threshold *`<threshold_value>`*

Specifies that only those transformations be committed whose score exceeds the threshold specified. *`<threshold_value>`* must be a valid float value between `0` and `1.0`, indicating the goodness of a transformation (as specified by `-objective`). Default is `0.0`.

Related Commands

define_memory_operation, insert_mem_clock_gating, insert_mem_observability_gating, insert_mem_sleep_gating, report_memory_gating

Usage

The `insert_mem_stability_gating` command creates a file `enLogic.v/vhd` in the PowerPro work directory. This file contains information about enable logic added by PowerPro to shut off redundant reading of memories.

Invoking the `insert_mem_stability_gating` command with no options is equivalent to invoking it as follows:

```
insert_mem_stability_gating  -effort medium  -objective efficiency
-max_area_limit 0.5  -threshold 0.0
```

In other words, sequential optimizations are performed with medium effort to maximize the efficiency metric using only stability based sequential transformations, where the additional area from the enable logic will not exceed 50% of the original design area, and all transformations whose goodness value exceeds 0.0 will be committed.

Examples

Example 1: Performs stability based clock -gating for those transformations whose score exceeds 0.2, at the same time ensuring that the increase in area as a result of the additional enable logic does not exceed 5% of the total design area.

```
insert_mem_stability_gating  -threshold 0.2  -max_area_limit 0.05
```

insert_observability_logic

insert_observability_logic — Performs observability based sequential optimizations to shut off loading of registers on the current design to reduce dynamic power. Note that this command can only be called after `prototype_design`.

Synopsis

```
insert_observability_logic [ -effort low | medium | high ]
    [ -objective area | efficiency ] [ -allow_flop_slicing ]
    [ -max_area_limit <area_limit> ] [ -threshold <threshold> ]
    [ -logic_depth <ldepth> ] [ -bounded ] [ -max_support <size> ]
    [ -optimal_expressions ]
```

-effort low | medium | high

Specifies the relative amount of CPU time to be spent during sequential optimization. Valid values are `low`, `medium` and `high`. Default: `medium`.

-objective area | efficiency

Specifies the objective that is to be maximized during observability-based sequential optimization.

If `area` is specified as the objective, `insert_observability_logic` commits only those sequential transformations that minimize the increase in area. Likewise, if `efficiency` is specified as the objective, `insert_observability_logic` commits only those transformations that maximize the efficiency metric of the design. Default: `efficiency`.

-allow_flop_slicing

Forces PowerPro to compute distinct observability conditions for each slice of a flop, when available.

This option will be deprecated in the next release, use the global `seq_opt_enable_cgobs_slicing` instead.

-max_area_limit *`<area_limit>`*

Ensures that the area increase, as a result of the additional enable logic, is within the limit specified by `area_limit`.

`area_limit` is a float value representing the fraction of the initial area of the design. The default is `0.5`, indicating that the area increase as a result of the additional enable logic must not exceed 50% of the original design area.

-threshold *`<threshold>`*

Commits only those transformations whose score exceeds the value specified by `threshold`. The score of a transformation is a float value between `0` and `1.0` indicating the goodness of the transformation. It is based on the objective that the user provides. The `threshold` value is a float representing a valid score. Default: `0.0`.

-logic_depth *`<ldepth>`*

Ensures that the core enable logic depth does not exceed the value specified by `ldepth`. Logic depth is defined as the depth of the logic when implemented using 2 -input gates. By default, logic depth is unconstrained.

-bounded

If this option is provided, unlike the default behavior of optimizing the enable logic to the optimal logic that will give maximum power savings, the tool will stop optimizing the logic once it meets the -logic_depth and -max_support constraints. All the moves optimized in the above fashion will be implemented irrespective of its power savings/loss as long as the -max_area_limit is met. -threshold is not compatible with this optimization mode and the tool will error out if it is provided.

-max_support *`<size>`*

Specifies the maximum number of bit -level inputs for the core enable logic. By default, the size is unconstrained.

-optimal_expressions

Enables the expression weakening flow to generate simpler expressions.

Related Commands

insert_stability_logic

Usage

Invoking the `insert_observability_logic` command with no options is equivalent to invoking it as

```
insert_observability_logic  -effort medium  -objective efficiency
-max_area_limit 0.5  -threshold 0.0
```

In other words, sequential optimizations are performed with medium effort to maximize efficiency increase using only observability based sequential transformations, where the additional area from the enable logic will not exceed 50% of the original design area, and all transformations whose goodness value exceeds 0.0 will be committed.

Information about the enable logic added by PowerPro to shut off redundant loading of registers is written to the *<work_directory>*/`enLogic.v (or enLogic.vhd)` file. For more information on the `enLogic.v/enlogic.vhd` file, refer to the PowerPro User Manual.

Examples

Example 1: Performs observability based sequential clock -gating for those transformations whose score exceeds 0.2 until the area consumed by the additional enable logic does not exceed 5% of the total design area.

```
insert_observability_logic  -threshold 0.2  -max_area_limit 0.05
```

insert_stability_logic

insert_stability_logic — Performs stability based sequential optimizations to shut off redundant reads of registers to reduce dynamic power. Note that this command can only be called after `prototype_design`.

Synopsis

```
insert_stability_logic [-type<typeStr> ]
     [ -effort low | medium | high ]  [ -objective area | efficiency ]
     [ -max_area_limit <area_limit> ] [ -threshold <threshold> ]
```

-type *<typeStr>*

> typeStr can be either C (Constant mode) or S (Symbolic mode). The default is type C followed by type S.

-effort low | medium | high

> Specifies the relative amount of CPU time spent during the sequential optimization. Valid values are `low`, `medium` and `high`. The default is `medium`.

-objective area | efficiency

Specifies the objective that is to be maximized by the sequential optimization. If the specified objective is `area`, the command commits those sequential transformations that minimize the area increase. If the objective is `efficiency`, the command commits those transformations that maximize the efficiency metric of the design. The default objective for stability based sequential optimization is `efficiency`.

-max_area_limit *<area_limit>*

Ensures that the area increased by the additional enable logic is smaller than the limit. The `area_limit` option is a float representing the fraction of the initial area of the design. The default is 0.5, meaning the area of the additional enable logic will not exceed 50% of the original design area.

-threshold *<threshold>*

Commits those transformations whose score exceed `threshold`. The score of a transformation is a float between 0 and 1.0 indicating the goodness of the transformation. It is based on the objective that the user provides. The `threshold` value is a float representing a valid score. The default is 0.0.

Related Commands

insert_observability_logic

Usage

The `insert_stability_logic` command creates a `<`*`workdir`*`>/enLogic.(v/vhd)` file with information about enable logic added by PowerPro to shut off redundant reading of registers. For more information about the `enlogic.(v/vhd)` file, refer to the PowerPro User Manual.

Invoking the `insert_stability_logic` command with no options is equivalent to invoking it as:

```
insert_stability_logic -effort medium -objective efficiency
    -max_area_limit 0.5 -threshold 0.0
```

In other words, sequential optimizations are performed with medium effort to maximize the efficiency metric using only stability based sequential transformations, where the additional area from the enable logic will not exceed 50% of the original design area, and all transformations whose goodness value exceeds 0.0 will be committed.

Examples

Example 1: Performs stability based sequential clock-gating for those transformations whose score exceeds 0.2 until the area consumed by the additional enable logic does not exceed 5% of the total design area.

```
insert_stability_logic -threshold 0.2 -max_area_limit 0.05
```

insert_static_signal_gating

Deprecated w.e.f. PowerPro 10.4.

insert_static_signal_gating — Generates static signal-based gating expressions.

Synopsis

```
insert_static_signal_gating
```

Related Commands

report_static_signal_gating, report_enable_expression, set_static_signal

Usage

The `insert_static_signal_gating` command generates static signal-based gating expressions for signals specified by the `set_static_signal` command.

A sample flow is shown below.

```
prototype_design
...
//Specifies the signals to be used during Static Signal-Based Gating
set_static_signal {sig1 sig2 sig3}

//Generates gating expressions based on the static signals sig1, sig2
and sig3
insert_static_signal_gating

//Returns only those gating expressions generated based on the
//static signals 'sig1', 'sig2 and 'sig3'.
//Same as specifying report_enable_expression  -static
report_enable_expression  -all
...
//Call Sequential Optimization Commands
<sequential_optimization_commands>

//Returns gating expressions generated based on the static signals
// 'sig1', 'sig2 and 'sig3' as well as the domains committed
//during sequential optimization
report_enable_expression  -all
```

Example

Example 1: Specifies that the signals `sig1, sig2` and `sig3` be used for generating gating expressions during Static Signal-Based Gating.

```
set_static_signal {sig1 sig2 sig3}
insert_static_signal_gating
```

limit

limit — Sets upper limits on resource usage for a PowerPro run.

Synopsis

`limit [ -time <value>] [ -memory <value>]`

-time *<value>*

Limits the clock time usage for this PowerPro run.

-memory *<value>*

Limits the memory usage for this PowerPro run.

Usage

The **limit** command sets upper limits on resource usage for a PowerPro run. The limit is effective from the invocation of the `limit` command, and is in addition to resources already consumed.

Typical values for time limits are: 45 (assumed to be seconds), 45s, 30min, 30m, 7hours, 7h, 2days, 2d. Typical values for the memory limit are: 900 (assumed to be MB), 900mb, 2gb.

If the specified `value` is 0 then any previously specified limit is removed. If no option is specified, then the current limit setting or settings and the remaining allowed resource quantity or quantities are printed.

Multiple `limit` commands can be used in the same PowerPro run. The most recent command applies if multiple commands set the same limit.

Examples

Example 1: Limits the permitted CPU usage of the current PowerPro session to 5 seconds.

```
limit  -cputime 5
```

link_design

link_design — Elaborates and links a design library.

Synopsis

```
link_design [ -top <top_module> ]
     [ -defparam <list_of_parameter_value_pairs> ]
```

-top `<top_module>`

Specifies the top module or entity of the design. A top module is a module that is not instantiated by any other module in the design. PowerPro requires the `-top` option to be explicitly specified when multiple top modules exist in the design.

-defparam `<list_of_parameter_value_pairs>`

Specifies the list of parameter value pairs to override default values for the parameters of the top level module.

-mem

The `-mem` option has been deprecated w.e.f PowerPro 7.1. Memory wrappers with the `*calypto_memory_view*` pragma are now recognized by default. This option no longer needs to be explicitly specified.

Related Commands

build_db, build_design, read_design

Usage

The `link_design` command elaborates and links one or all of the design libraries to resolve definitions of module instances. A design must be linked before it can be built.

Notes

- Modules can be referred to within other module definitions.
- A design library cannot have multiple definitions of the same module.
- The active design library is searched during linking to resolve the reference. If the module definition is not found, an error message is displayed.

Examples

Example 1: Reads in the design file `input.v` and then links the design libraries.

```
read_design input.v
link_design
```

Example 2: Overrides the values of the parameters `p1`, `WIDTH` and `val` in the top module `input` of the design and then links the design libraries.

```
read_design input.v
link_design -defparam "{p1 1} {WIDTH $width} {val "\"FIRST\""}" -top
input
```

load_upf

The `load_upf` command has been deprecated w.e.f. PowerPro 10.2. To set the path to the UPF file, use the global `upf_file_name` and to specify the version of the UPF file, use the global `upf_version`.

load_upf — Reads a Unified Power Format (UPF) file. The supported UPF versions are UPF 1.0, UPF 1.1, UPF 2.0, and UPF 2.1.

Synopsis

`load_upf <filename>`

`<filename>`

Specifies the name of the UPF file that must be read.

Related Commands

set_voltage

Usage

The **load_upf** command is used to read the power intent, specified in a UPF file. The commands in the UPF file are read in sequence and constraints are applied as and when required. The `load_upf` command must be run only after the global `_new_fe_enable` is enabled and before the `build_design` command.

In PowerPro, sequential optimization can be used to support designs with multiple power domains operating at different voltages. The design is partitioned into multiple power domains – each operating at different voltages using the UPF file. In such cases, PowerPro honors the voltage applicable through the UPF file while performing power analysis.

Note: This command is under limited production support.

Example

Example 1: Reads the UPF file `rtl_top_1.upf`.

```
load_upf rtl_top_1.upf
build_design
```

mark_shift_register

mark_shift_register — Detects and marks register chains in the design. All flops in design matching the register chains are marked with the dont-optimize constraint and signals between such flops are marked with the dont-use constraint. Note that this command can only be called after `prototype_design`.

Synopsis

```
mark_shift_register [ -flop_stage <depth> ] [ -filename <file_name> ]
```

-flop_stage *`<depth>`*

Specifies the depth of register chains to be identified. Default depth of register chains considered is `4`.

-filename *`<file_name>`*

Specifies the name of the file to which information about constraints applied to objects that match the shift -register chains are written to. Default file name is *`<workdir>`*`/shift_registers_constraints.tcl`.

Related Commands

mark_synchronizer

Usage

The **mark_shift_register** command identifies and marks shift -register chains belonging to the same clock domain in the design database. When a flop matching the criteria is found, it is marked with the dont-optimize constraint. Likewise, when a matching signal is found, it is marked with the dont-use constraint.

Examples

Example 1: Attempts to find a chain of 6 flops. Matching flops are marked with the dont-optimize constraint and matching signals with the dont-use constraint. Since the `-filename` option is not specified, constraints are written to the *`<workdir>`*`/shift_registers_constraints.tcl` file.

```
mark_shift_register  -flop_stage 6
```

Example 2: Attempts to find a chain of 8 flops. Matching flops are marked with the dont-optimize constraint and matching signals with the dont-use constraint. Constraints are written to the `shift_reg_constraint.tcl` file.

```
mark_shift_register  -flop_stage 8  -filename shift_reg_constraint.tcl
```

mark_synchronizer

mark_synchronizer — Detects and marks synchronizer flop patterns in the design database. Flops matching the specified pattern are marked with the dont-optimize constraint while signals between such flops are marked with the dont-use constraint. Note that this command can only be called after `prototype_design`.

Synopsis

```
mark_synchronizer [ -user_sync_cell ]
    [ -multi_flop_synchronize_scheme ] [ -input_to_flop_chain ]
    [ -common_enable_scheme ] [ -common_select_scheme ]
    [ -allow_comb_logic ] [ -flop_stage <depth> ]
    [ -ignore_inverter <inverter_count> ]
    [ -sync_reset < AND | OR | NAND | NOR > ]
    [ -control_signal_dest_domain ] [ -filename <file_name> ]
```

-user_sync_cell

Specifies that all flops within modules specified using the `set_existing_synchronizer` command be marked as synchronizer flops.

-multi_flop_synchronize_scheme

Specifies that all flops matching the multi-flop synchronizer scheme be marked with the dont-optimize constraint while signals between such flops be marked with the dont-use constraint. This scheme is generally used to synchronize the control path.

-input_to_flop_chain

Specifies that all flops matching the multi-flop synchronizer scheme be marked even when the data signal to be synchronized is coming from a primary input or a blackbox output port. When this option is specified, the signal to be synchronized is marked with the dont-use constraint; flops matching the multi-flop synchronizer scheme are marked with the dont-optimize constraint and signals between such flops be marked with the dont-use constraint.

Note: This option expects the `-multi_flop_synchronize_scheme` option to be specified.

-common_enable_scheme

Specifies that all flops matching the `common enable synchronizer scheme` be marked with the dont-optimize constraint while signals between such flops be marked with the dont-use constraint. This scheme is generally used to synchronize the data path.

-common_select_scheme

Specifies that all flops matching the `common select synchronizer scheme` be marked with the dont-optimize constraint while signals between such flops be marked with the dont-use constraint. This scheme is generally used to synchronize the data path.

-allow_comb_logic

The behavior of this option depends on the type of synchronizer pattern specified.

1. When specified for the multi flop synchronizer pattern, combinational logic is allowed on the data transfer path.

2. When specified for the common enable synchronizer pattern, combinational logic is allowed on the enable path of the last flop.

3. When specified for the common select synchronizer pattern, combinational logic is allowed on the select path of the selector -mux.

Note that the `-ignore_inverter` option is ignored for paths where the `-allow_comb_logic` option is applicable.

-flop_stage *`<depth>`*

Specifies the depth of the synchronizer flop in the multi flop synchronizer scheme. Note that this option is honored only when the `-multi_flop_synchronize_scheme` has been specified. Default is `2`.

-ignore_inverter *`<inverter_count>`*

Specifies the number of inverters allowed in the path between synchronizer flops in different synchronizer schemes. By default, there is no limit to the number of inverters that can appear on the paths.

-sync_reset *`< AND | OR | NAND | NOR >`*

The behavior of this option depends on the type of synchronizer pattern specified. Note that this option is honored only when the `-multi_flop_synchronize_scheme` or the `-common_select_scheme` option has been specified.

When specified with the `-multi_flop_synchronize_scheme`, a node with the specified node type is allowed just before all synchronizer flops.

When specified with the `common select scheme`, a node with the specified node type is allowed after the selector-mux node.

-control_signal_dest_domain

This option is honored only when the `-common_select_scheme` option or the `-common_enable_scheme` option is specified.

When specified with the `-common_select_scheme` option, a single flop is matched instead of matching the synchronizer on the select of the mux.

When specified with the `-common_enable_scheme` option, a single flop is matched instead of matching the synchronizer on the enable of synchronizer flop.

-filename *<file_name>*

Specifies the name of the file to which all constraints applied to objects that match the specified synchronizer pattern, are written to. By default, the constraints are written to the *<workdir>*`/synchronizer_flops.tcl` file.

Related Commands

set_existing_synchronizer

Usage

The **mark_synchronizer** command detects and marks synchronizer flop patterns in the design database. Flops matching the specified pattern are marked with the dont-optimize constraint while signals are marked with the dont-use constraint.

Examples

Example 1: Searches for the double flop synchronizer pattern in the design and marks all matching flops with the dont-optimize constraint. Constraints are written to the *<workdir>*`/synchronizer_flops.tcl` file.

```
mark_synchronizer  -multi_flop_synchronize_scheme
```

opt_infer_and_set_sync_resets

opt_infer_and_set_sync_resets — This command provides the synchronous reset awareness in the PowerPro Optimizer flow. It generates potential synchronous resets from the design and `set_powerpro_reset` constraints corresponding to these potential resets for different clock domain and polarity in the `opt_infer_sync_resets.tcl` file. If the `–port` option is used, it marks all inferred resets, avoids the redundant moves from the inferred resets, and performs sync reset aware patching in the downstream optimization flow.

Synopsis

`opt_infer_and_set_sync_resets [-port <`*`top-level port`*`>]`

-port <*top-level port*>

> Marks all inferred resets and avoids the redundant moves from the inferred resets in the downstream optimization flow.

Related Commands

set_powerpro_reset

Usage

When the `opt_infer_and_set_sync_resets` command is run, the suggestion file `opt_infer_sync_resets.tcl` is generated, which has top-level `set_powerpro_reset` constraints for different clock domains and polarity. This file can be used as a reference to explicit specify PowerPro reset constraints before STB optimization.

Example

Example 1: Performs an analysis of the design file `input.v` to identify and mark the sync resets.

```
build_design input.v
prototype_design
opt_infer_and_set_sync_reset –port <top-level port>
```

parse_and_mark_timing_critical_paths

parse_and_mark_timing_critical_paths — Disables PowerPro gating using the timing critical path information specified in the Timing Critical Report. Note that this command can only be called after `prototype_design`.

Synopsis

```
parse_and_mark_timing_critical_paths
    -filename <timing_report>  -tool_type <timing_report_type>
    [ -slack_threshold <threshold_value> ] [ -use_name_fallbacks ]
    [ -suffix <suffix_string> ] [ -output_file_name <file_name> ]
    [ -apply_constraints ]
```

-filename *<timing_report>*

Specifies the path to the Timing Critical Report. In case, reports are available at multiple locations, specify this option multiple times.

-tool_type *<timing_report_type>*

Specifies the name of the tool that generated the Timing Critical Report. Timing reports generated by RC and DC are supported. Legal values are: `RC` and `DC`.

-slack_threshold *<threshold_value>*

Specifies the slack threshold value. *<threshold_value>* must be a float or integer value. In case this option is not specified, the default value of 0.00 is used as the slack threshold. Note that constraints will be generated only when the slack threshold is greater than the path slack.

-use_name_fallbacks

Enables the name fallback mechanism to match report names with PowerPro names.

-suffix *<suffix_string>*

Specifies that the suffix be removed before matching names. This option requires the `-use_name_fallbacks` option.

-output_file_name *<file_name>*

Specifies the name of the output file. In case an absolute/relative path is not provided, the file is generated in the current working directory.

In case this option is not specified, a file by the name of `default_constraints_file.tcl` is generated in the current working directory.

-apply_constraints

Specifies that constraints be automatically applied after they are generated.

Related Commands

set_dont_use

Usage

The **parse_and_mark_timing_critical_paths** command parses timing reports that have timing critical path information to identify constraints that need to be applied during gating. If specified, a constraints file is also generated. This file is then sourced by PowerPro, to apply constraints, before gating.

Examples

Example 1: Generates a Tcl file, by the name of *<ppro_gen_design_constraints.tcl>*, containing all flops marked `set_dont_optimize` and all signals marked `set_dont_use` that fall within the timing critical path. By specifying the `-apply_constraints` option, the constraints are not only generated but applied as well.

```
build_design input.v
...
...
prototype_design
parse_and_mark_timing_critical_paths  -filename rc_timing_report.txt
     -tool_type RC  -slack_threshold  -2.00  -use_name_fallbacks
     -suffix "_reg"  -output_file_name ppro_gen_design_constraints.tcl
     -apply_constraints
```

pget

pget — Accesses the entities and respective attributes of the PowerPro-generated database.

Synopsis

```
pget [-u ] [-v ] [-s ] [-a ] [-format [tcl | line | column] ] [-dfs
    [fanin | fanout] ] [-visit [insts | signals | ports] ] [-depth
    <depth>] [-skip_insts] [-skip_signals] [-skip_ports] [-filename]
```

-u

Returns only the unique objects.

-v

Returns only those objects that do not match the wild card pattern.

-s

Exits the command without any error, if an empty list of objects is passed to the `pget` command. An empty list may be created during chaining of `pget` commands.

-a

Applies multiple attributes simultaneously on the specified `pget` objects.

-format [tcl | line | column]

Specifies the format in which the output must be generated. By default, the output is generated in `tcl` format, the other two being `line` and `column` formats.

In `tcl` format, the output is printed in the form of a tcl list. The tcl list can be used to create a tcl script.

In `line` format, every object is printed in a new line.

In `column` format, the output is printed in a column-separated format.

-dfs [fanin | fanout]

Specifies the depth-first-traversal in the `fanin` or `fanout` direction in the PowerPro database. The `dfs` option starts with an object pointer or a list of object pointers and generates a list of objects visited in the direction specified. The object that must be visited during traversal can be specified using the `visit` option.

-visit [insts | signals | ports]

Specifies the type of object that must be visited during depth-first-traversal.

This option can be used only with the `dfs` option.

-depth <depth>

Specifies the maximum depth till which the depth-first-traversal must visit the objects of type specified by the `visit` option. Possible values are positive integers greater than `1`.

The `depth` option is optional and can be used only with the `dfs` option.

-skip_insts

Specifies the attributes of the instance until which the depth-first-traversal must visit the objects of the type specified by the `visit` option.

The `skip_insts` option is optional and can be used only with the `dfs` option.

-skip_signals

Specifies the attributes of the signal until which the depth-first-traversal must visit the objects of the type specified by the `visit` option.

The `skip_signals` option is optional and can be used only with the `dfs` option.

-skip_ports

Specifies the attributes of the port until which the depth-first-traversal must visit the objects of the type specified by the `visit` option.

The `skip_ports` option is optional and can be used only with the `dfs` option.

-filename

Stores the output of the `pget` command in a text file.

Related Commands

all_inputs, all_outputs, all_registers

Usage

The command `pget` is used to access the entities and respective attributes of the PowerPro-generated database and generate their text reports. The command primarily works by chaining of attributes. It either returns pointers or end results. The pointers can be further used to chain the attribute to get the end result.

There are only two top-level attributes: `design` and `libraries`.

```
powerpro> pget help
```

1. design: Returns design (top design hier) object pointer.

2. libraries: Returns object pointer of the libraries read in the PowerPro.

These are further associated with a list of attributes that can be displayed by appending `.help` to these top-level attributes as shown below or to the attribute chain as shown in example 11 below.

```
powerpro> pget libraries.help
```

Attributes for libraries:

1. name: Returns name of the library.

2. path: Returns path of the library.

3. cells: Returns cell pointers in the library.

4. opercond: Returns operating condition pointer of the library.

Examples

Example 1: Returns the names of all the instances in the design.

```
pget design.insts.name
```

Example 2: Returns the pointers to all the ports that are present in the design. This is an intermediate output, on which the command is further run to get the final output.

```
pget design.ports
```

Example 3: Returns the names of all the instances in the design that match `core1/ethernet_proto*`.

```
pget design.insts.name core1/ethernet_proto*
```

Example 4: Returns the pointers of all the instances whose names match with `core1/ethernet_proto*`. Note that the position of `_p` in the attribute chain specifies the attribute that must be printed. If `_p` is not appended to any of the attributes, then, by default, the last attribute in the command chain is printed.

```
pget design.insts_p.name core1/ethernet_proto*
```

Example 5: Returns the pointers of all the instances whose names match with `core1/ethernet_proto*`. The next `pget` command is then applied to get the names of the ports for these instance pointers. Note that the nesting of the `pget` commands allows to filter the results based on various attributes.

```
pget [pget design.insts_p.name core1/ethernet_proto*].ports.name
```

Example 6: Returns the output port names of all the instances whose names match with `core1/ethernet_proto*`.

```
pget [pget [pget design.insts_p.name
core1/ethernet_proto*].ports_p.isOp 1].name
```

Example 7: Returns those instances in the design for which the internal power is greater than `0.23`.

```
pget design.insts.power.internal > 0.23
```

Example 8: Returns the names of those instances that consume an internal power of more than `0.23`.

```
pget [pget design.insts_p.power.internal > 0.23].name
```

Example 9: Returns the unique cell names of all the instances of the design.

```
pget -u design.insts.cells.name
```

Example 10: Returns the unique cells of all the instances of the design whose names do not match with `*BUF*`.

```
pget -u -v design.insts.cells.name *BUF*
```

Example 11: Displays a list of attributes and their descriptions that are available for the specified attribute.

```
pget design.insts.help
```

Example 12: Displays the names and cell names of all the instances in the design whose names match with `*reg*`.

```
pget design.insts -a {name *reg*, cells.name}
```

Example 13: Displays the names, cell types, cell names, internal power, and switching power of all the instances in the design, where the names match with `*abc*`, the celltype is `register`, and the internal power is greater than `0.2`.

```
pget design.insts -a {{name *abc*, celltype register}, cells.name,
power{internal>0.2, switching}}
```

Example 14: Displays the names of all the instances in the design between the two given instances.

```
pget [pget design.insts.name a/b/reg1] -dfs fanin -visit insts -
skip_insts {name a/b/reg2}
```

Example 15: Stores the output of the `pget` command in the `xyz.txt` file, in the directory where powerpro is running. Note that it appends the `.txt` extension on its own.

```
pget design.hiers.ports -filename xyz
```

preserve_lib_module

preserve_lib_module — When optimizing the RTL for a given hierarchy, `preserve_lib_module` modifies the instantiation only, keeping the original content intact. The command can be called only before the `build_design` command.

Synopsis

```
preserve_lib_module -module <module> { -memory | -flop }
    [ -use_port <ports>] [ -optimize_with <new_module>]
    [ -assume_one <ports>] [ -assume_zero <ports>]
```

-module *<module>*

Specifies the name of the user hierarchy to preserve.

-memory

Specifies that the memory in the hierarchy be optimized.

-flop

Specifies that the flop-enable in the hierarchy be optimized.

-use_port *<ports>*

Specifies the name of the port to use when optimization finds more than one enable port for an opportunity.

-optimize_with *<new_module>*

When used with `-flop`, rebinds the original module with a different module in the resulting RTL modification.

-assume_one *<ports>*

When used with the `-flop` and `-optimize_with` options, assumes a default value of `1` on input port(s) that exist in the new module but not in the original module.

-assume_zero *<ports>*

When used with the `-flop` and `-optimize_with` options, assumes a default value of 0 on input port(s) that exist in the new module but not in the original module.

Related Commands

insert_mem_clock_gating, insert_mem_observability_gating, insert_mem_sleep_gating, insert_mem_stability_gating, insert_observability_logic, insert_stability_logic

Usage

The **preserve_lib_module** command constrains the tool from making changes inside flop library cells and memory wrappers. New enable logic will be moved outside the instantiation and the RTL modification will use the existing write-enable port, if any.

Optionally, the instance binding can be changed on-the-fly using the `-optimize_with` option. When specifying this option, the new module interface (ports) must be a superset of the original interface, as rebinding will only unplug the original connection and swap the new module with the modified RTL. If the `-assume_one` or the `-assume_zero` is not specified, any new input ports on the new module would be considered open and connected to `Z`.

Some designs have library modules for registers and memory wrappers that users would not like PowerPro to touch because of flow reasons. The `preserve_lib_module` constrains PowerPro from making changes inside such flop library cells and memory wrappers. Without additional information, PowerPro inserts gating logic in the module where the flop assignment is made or, in the case of memory gating, at the level where the memory is instantiated. Additionally the register or memory wrappers may be uniquified when their modifications vary from instance to instance.

Examples

Example 1: Assume that the user has a primitive for a flop (mapped to special cell in the library) as follows:

```
module DFFE_prim (clk, rst, d, en, q);
  parameter W = 8;
  input clk, rst, en;
  input [W-1:0] d;
  output reg [W-1:0] q;

  always @(posedge clk)
  begin
    if (rst)
      q <= {W {1'b0}};
    else if (en)
      q <= d;
  end
endmodule
```

Further assume that the design instantiates these flop primitives:

```
module TOP(input clk, sel, rst, input [15:0] in, output [15:0] out);
   wire [15:0] in_p1;
   wire sel_p1;
   DFFE_prim #(.W(16)) data_p1(clk, rst, in, 1'b1, in_p1);
   DFFE_prim #(.W(1)) ctrl_p1(clk, rst, sel, 1'b1, sel_p1);
   assign out = (sel_p1) ? in_p1 : 16'b0;
endmodule
```

The user needs to identify such library modules before the `build_design` is called. In the following script, `preserve_lib_module` optimizes the flop within `DFFE_prim` by connecting new enable logic to the enable pin. PowerPro is able to identify enable pins and optimizes the design by adding new controlling logic to the interface of `DFFE_prim`.

```
preserve_lib_module -flop -module DFFE_prim
build_design design.v -top TOP
create_clock -name CLK -period 1 clk
prototype_design
insert_observability_logic -effort high
write_rtl -output_dir ppro_rtl
```

Here's what the newly generated PowerPro RTL will look like:

```
module TOP(input clk, sel, rst, input [15:0] in, output [15:0] out);
   wire [15:0] in_p1;
   wire sel_p1;
   wire [0:0] cg_o_q_en; // PowerPro-CG
 //PowerPro-CG
   cg_obs_TOP inst_cg_obs_TOP( .rst( rst ),
         .d( sel ),
         .q_en( cg_o_q_en ));
 //DFFE_prim #(.W(16)) data_p1(clk, rst, in, 1'b1, in_p1);
 //PowerPro-CG
 DFFE_prim #(.W(16))
 data_p1(clk,
        rst,
        in,
        cg_o_q_en,
        in_p1);
   DFFE_prim #(.W(1)) ctrl_p1(clk, rst, sel, 1'b1, sel_p1);
   assign out = (sel_p1) ? in_p1 : 16'b0;
endmodule
```

Example 2: For various flow reasons, including insertion of BIST logic, it is possible that the user might want to insert memory-gating logic a level or two above the actual memory instance. To do so, issue the following command before calling the `build_design` command:

```
preserve_lib_module -memory -module RAM_sp_1024x32_wrapper
```

If there are multiple ports on the wrapper on which the gating logic can be pushed back on, the user needs to specify the desired ports to be pushed back on. Here is an example:

```
module RAM_sp_1024x32_wrapper(adr, d, clk, we, me, bme, biste, ls, q);

  wire me_int = (biste) ? me : bme;
  RAM_sp_1024x32 i_ram(adr, d, clk, we, me_int, biste, ls, q);
endmodule
```

In the above wrapper, ME-based gating logic can be pushed back on `me`, `bme` and `biste` ports of `RAM_sp_1024x32_wrapper`. But the user would like the gating to be pushed back on the `me` pin. In such a situation, user would provide the additional option of `-use_port` to instruct the tool not to push back on `biste` or `bme` pin. For light sleep-based gating logic, the user must specify the light sleep pin as well.

```
preserve_lib_module -memory -module RAM_sp_1024x32_wrapper -use_port
{me ls}
```

When the `-use_port` option is specified, all other ports are excluded from modification. It is important that the user provide a complete list, as done for the ports `me` and `ls`.

prototype_design

prototype_design — Normalizes the design and performs generic optimization. Prepares the design for sequential and power analysis. Note that this command can only be called after `build_design`.

Synopsis

`prototype_design`

-power

> *The `-power` option has been deprecated w.e.f PowerPro 7.1. The default flow now includes support for the Power Analysis mode. This option no longer needs to be explicitly specified.*

Related Commands

build_design

Usage

The **prototype_design** command performs normalization and logic optimization. The following steps are performed as part of `prototype_design`.

Consistency checks	Ensures that the library, netlist and constraints provided to PowerPro are consistent and are within the valid subset accepted by the tool.
Normalization	Transforms the design to satisfy database invariants. Since the design is stored in the database at the operator level, certain transformations like "dissolving false word-level loops" are required.
Generic Optimizations	Performs technology independent optimizations to remove redundancy in the design.

Examples

Example 1: Transforms the design and prepares it for observability based sequential optimization, as indicated by the `insert_observability_logic` command in the flow below.

```
read_library techlib.lib
build_design input_rtl.v
create_clock -name CLK clk -period 2
prototype_design
insert_observability_logic
```

read_db

read_db — Reads and restores a built database.

Synopsis

`read_db <file>`

`<file>`

File containing a previously written database.

The **read_db** command reads and restores a previously written database.

Usage

The **read_db** may be used instead of reading designs through `build_design`.

The `read_db` cannot *not* be used in combination with `build_design` command.

Examples

Consider two PowerPro runs: One which writes the database just before the stage where tool is ready to run Sequential optimization. The other one read in the DB file from the first PowerPro run, and continues with the regular PowerPro flow (Sequential optimization in this case).

Following is first PowerPro run where we write out the DB file.

```
build_design rtl_main.v rtl2.v rtl3.v rtl4.v
prototype_design
set_cg_override  -port in
write_db pre_optimization.db
exit
```

In second PowerPro run, we read in the DB file and continue with the flow, insert_observability_logic in this example.

```
read_db pre_optimization.db
insert_observability_logic
write_rtl
write_verify_script
```

read_design

read_design — Reads design files into the PowerPro design library.

Synopsis

```
read_design [ -verilog | -vhdl | -sv ] <arguments>...
```

-verilog | -vhdl | -sv

Specifies the design language of the files being read in. If omitted, the extension of the first file read in is used to determine the design language (Verilog if `.v`, VHDL if `.vhd` or `.vhdl`, and SystemVerilog if `.sv`).

Verilog and SystemVerilog Arguments

+define+*<macro>* [=*<macro_body>*]

Defines a macro. For example, to assign the value `1024` to the macro `WIDTH` when the file `input.v` is read in, specify the following command:

```
read_design +define+WIDTH=1024 input.v
```

-f*<list_file>*

Specifies the name of the file which contains the list of files and options to be read in.

-y*<dir_name>*

Specifies that PowerPro search for unresolved modules in the directory *<dir_name>*.

-v*<file_name>*

Specifies that PowerPro search for unresolved modules in the file *<file_name>*.

+libext+*<ext>*

Specifies that PowerPro search for unresolved modules in the library directory with extension *<ext>*.

+librescan

Specifies that PowerPro search from the beginning of the library list for undefined modules, and force multiple searches in the same library.

+incdir+*<dir>* ...

Specifies that PowerPro search + separated directories to resolve `` `include `` directives.

<filenames> ...

Specifies a whitespace separated list of design files to be read into PowerPro which might include absolute or relative directory paths. By default, any additional file references included within the design files are resolved within the current working directory. Additional include paths can be specified using the `+incdir+`*`<dir>`* option.

VHDL Arguments

-library *<library>*

Specifies that the VHDL units being read in belong to the VHDL logical library *<library>*.

Related Commands

create_black_box, link_design

Usage

The `read_design` command reads in a set of design files. Note that the design must be linked and built using the `link_design` and `build_db` commands after reading it in.

Notes

- All files read in using a single `read_design` command must be in the same language. For example,:

```
read_design -verilog input1.v input2.v
read_design -vhdl input3.vhd
```

- Multiple `read_design` commands can be issued for a design until it is linked.
- If a module contains non-synthesizable code, the module should be blackboxed before it is read in. See the `create_black_box` command for more information.
- Any remaining command arguments are passed as options to the language-specific reader. Options not understood by the reader might be ignored or, in some cases, might also generate an error.

Examples

Example 1: Reads in the Verilog design files `input1.v` and `input2.v`.

```
read_design -verilog input1.v input2.v
```

Example 2: Reads in the SystemVerilog file `design.v`.

```
read_design -sv  design.v
```

Example 3: Sequentially reads in design files in different languages. For the first `read_design` command, the files are assumed to be written in Verilog based on the file extension.

```
read_design input1.v input2.v
read_design -vhdl input3.vhd
read_design -sv input4.v
```

read_eco_db

read_eco_db — Reads files for Engineering Change Order (ECO) flow.

Synopsis

```
read_eco_db <file>
```

`<file>`

Specifies the name of the pre-ECO design database to be read. This is the design database that includes the optimizations generated by PowerPro.

Related Commands

set_powerpro_mode

Usage

The **read_eco_db** command reads in the Pre-ECO database file. This design database must have been created using the `write_db` command, executed at the end of sequential optimization, in the pre-ECO PowerPro run.

This command can only be called after reading the technology libraries and before reading the ECO design. Additionally, the PowerPro mode must be set to `eco`.

If there are no moves during PowerPro optimization, the user can skip SlecPro verification or optimization.

Examples

Example 1: Reads in the pre-ECO design database file.

```
powerpro> set_powerpro_mode eco
powerpro> read_eco_db pre_eco_run.db
```

read_fsdb

read_fsdb — Reads the switching activity data file that is generated during simulation. To read an FSDB file, set the UNIX shell environment variable `NOVAS_LD_LIBRARY_PATH` to specify the directory location that contains the latest version of the dynamic library files `libnffr.so` and `libnsys.so`. 5.8 and earlier are supported. Note that the 2019.06 version of FSDB reader must be used with PowerPro 10.4.

Synopsis

```
read_fsdb <filename> -instance_name <hierarchical_instance_name>
    [-use_name_fallback { on | off }]
    [-complex_name_matching { on | off }]
    [-fsdb_name_suffix <suffix_string> ]
    [-begin <begin_time> ] [-end <end_time> ]
    [-print_hierarchy]
    [-weight <weight>][-zero_delay]
```

<filename>

Specifies the name of the FSDB file.

-instance_name *<hierarchical_instance_name>*

Specifies the hierarchical name of the top module of the design, as appearing in the FSDB file. Note that this option must not be specified when specifying the –`print_hierarchy` option.

-use_name_fallback { on | off }

When enabled, if a signal name obtained from the FSDB file does not match any signal name in the design, the name of the signal obtained is modified to remove escape characters and indices before attempting a match again. Default is `on`.

-complex_name_matching { on | off }

When enabled, PowerPro Matches complex names like generate-block names from the FSDB file. Default is `on`.

Note: While enabling this option leads to better switching activity assertion, it could also lead to increased run time.

-fsdb_name_suffix *<suffix_string>*

Specifies that the names in the FSDB file contain a suffix *<suffix_string>* that must be removed before matching a signal or instance name obtained from the FSDB file to a signal or instance name in the design.

-begin *<begin_time>*

Specifies the start time of simulation window from FSDB. If the time unit is not specified, PowerPro automatically infers it from the FSDB file. Supported time units are: `s, ms, us, ns, ps, fs, as, za and ys.`

-end *<end_time>*

Specifies the end time of simulation window from FSDB. If the time unit is not specified, PowerPro automatically infers it from the FSDB file. Supported time units are: `s, ms, us, ns, ps, fs, as, za and ys`.

-print_hierarchy

Displays the hierarchical structure of the design for which the FSDB file was created. This option expects the FSDB filename *<filename>* to be specified. When this option is specified, all other options are ignored.

-weight <*weight*>

Specifies the value by which switching activity data in the FSDB file must be scaled before annotating on the switching activity data. Legal values are float values `>0` and `<=1`.

Note: It is not recommended to create apps and queries in PowerPro Designer if multiple fsdb files with the `-weight` option are used in the flow.

-zero_delay

Specifies that the FSDB file contains no delay information, and power computation will happen using Zero Delay Power Computation Engine. When this option is not specified, PowerPro will assume that the FSDB file has delay information and will use Delay Aware Power Computation Engine. Note that this option will only have effect on GLPA flow with the `-pa` option.

Related Commands

read_saif

Usage

The **read_fsdb** command specifies the FSDB file to be used during sequential optimization. Switching activity is automatically inferred from the FSDB file.

If multiple `read_fsdb` commands are issued, the last specified `read_fsdb` command overrides previously issued commands. In this example, the simulation engine will use the FSDB file `design2.fsdb`.

```
read_fsdb design1.fsdb -instance_name testbench.dut
read_fsdb design2.fsdb -instance_name testbench.dut
```

Note: When the `-print_hierarchy` option is specified, the `read_fsdb` command does not read the FSDB file but only prints the hierarchy of the FSDB file.

Examples

Example 1: Reads the FSDB file `design.fsdb`. During simulation, only those signals which fall within the scope of the instance `testbench.dut` will be considered. Here, instance name `testbench` is the name of the testbench module while `dut` is the name of the instance corresponding to the top module of the design within the testbench.

```
read_fsdb design.fsdb -instance_name testbench.dut
```

Example 2: Reads the FSDB file `design.fsdb` and annotates those signals that are in the scope of the instance `testbench.dut` within the simulation window `1000000000us` and `1010000000us`. In this example, since the time unit is not specified, the time unit is automatically inferred from the FSDB file.

```
read_fsdb -instance_name testbench.dut design.fsdb -begin 1000000000 -
end 1010000000
```

Example 3: Reads the FSDB file `design.fsdb` and annotates the signals in the scope of the instance `testbench.dut` within the simulation window `1000000000000` to `4000000000000` on the fsdb scale `1fs`. In this example, while the FSDB scale is in `fs`, the user specifies the time unit in the `read_fsdb` command as `ms`. In such cases, PowerPro automatically converts the time specified in the `read_fsdb` command to use the time scale specified in the FSDB file.

```
read_fsdb -instance_name testbench.dut design.fsdb -begin 1ms -end 4ms
```

Example 4: Prints the hierarchy of the FSDB file `design.fsdb`. Results of the command are also shown below.

```
powerpro> read_fsdb -print_hierarchy design.fsdb

TESTE
TESTE.DUT_INST_0
TESTE.PROCESS_0
TESTE.PROCESS_1
TESTE.PROCESS_2
TESTE.PROCESS_3
TESTE.DUT_INST_0.PROCESS_0
```

Example 5: This example illustrates the usage of the -weight option with the read_fsdb commands.

```
powerpro> read_fsdb -weight 0.1 fsdb_1.fsdb -instance_name
testbench.dut
powerpro> read_fsdb -weight 0.4 fsdb_2.fsdb -instance_name
testbench.dut
powerpro> read_fsdb -weight 0.5 fsdb_3.fsdb -instance_name
testbench.dut
```

will annotate switching activity information as follows:

SA(n1) = (0.1*fsdb_1 + 0.4*fsdb_2 + 0.5*fsdb_3)

read_library

read_library — Reads the technology library files in Liberty (`.lib`) format or in compressed formats. Supports parser version 2.6. Note that this command can be called only before the `build_design` command.

Synopsis

```
read_library {<library_list>}
```

`{<library_list>}`

Specifies the path to the files that contain the technology data.

Related Commands

bind_to_tech_cell, find_lib_cell, set_binding, set_default_threshold_voltage_group, set_operating_condition, set_target_technology, set_threshold_voltage_group

Usage

The **read_library** command reads the technology library files in Liberty format. The `read_library` command can read in both `.lib` files as well as technology library files in compressed formats. Supported compressed formats include `.zip`, `.gz`, `.Z` and `.bz2`.

Apart from this, PowerPro has a utility called `clib` to create an encrypted version of the user library. The encrypted library can be read in PowerPro, just like a regular liberty file, using the `read_library` command. For more information on using the `clib` utility, refer to the *PowerPro User Manual*.

Examples

Example 1: Reads in the library file `tsmc65.lib`.

```
read_library tsmc65.lib
build_design rtl.v
prototype_design
```

Likewise, multiple library files (`.lib`) can be read with a single command.

```
read_library {A.lib B.lib C.lib tsmc65.lib}
```

Example 2: Reads in the compressed library file `sample.lib.gz`.

```
read_library sample.lib.gz
```

To read multiple compressed files a single `read_library` command can be issued as follows.

```
read_library { sample.lib.gz simple.lib.gz }
```

Example 3: Reads in a combination of compressed files and `.lib` files.

```
read_library { sample.lib.gz sample1.lib }
```

read_name_map_file

read_name_map_file - Reads the RTL to Gate Level netlist name-map file. The name-map file contains a list of `set_rtl_to_gate_name` commands. Note that this command must be called after the `build_design` command and before reading the simulation file (such as, `read_fsdb`) or the SPEF file (such as, `read_spef`), and the global `pa_rtl_sim_on_glpa` must be enabled in case of RTL sim on GLPA flow.

Synopsis

```
read_name_map_file <name_map_file> [-instance <instance_name>] [-
    rtl_instance <rtl_instance_name>] [-rtl_hierarchy_separator <
    rtl_hierarchy_separator >] [-gate_level_hierarchy_separator
    <gate_level_hierarchy_separator>] [-spef]
```

`<name_map_file>`

Specifies the name of the file that contains a list of the `set_rtl_to_gate_name` commands.

-instance `<instance_name>`

Specifies the hierarchical name of the top module, which appears as part of the `-rtl` and `-gate` options of `set_rtl_to_gate_name` commands listed in the `name_map_file` file.

-rtl_instance `<rtl_instance_name>`

Specifies the hierarchical name of the top module, which appears as part of the `-rtl` option of `set_rtl_to_gate_name` commands listed in the `name_map_file` file. If this option is not specified, the value of `rtl_instance_name` option is automatically read from the `-instance` option.

-rtl_hierarchy_separator <*rtl_hierarchy_separator*>

Specifies the hierarchical separator for the RTL name, which is provided with the –`rtl` option of `set_rtl_to_gate_name` commands listed in the `name_map_file` file. If this option is not specified, the hierarchy separator / is used by default.

-gate_level_hierarchy_separator <*gate_level_hierarchy_separator*>

Specifies the hierarchical separator for the gate-level name, which is provided with the `-gate` option of `set_rtl_to_gate_name` commands listed in the `name_map_file` file. If this option is not specified, the hierarchy separator / is used by default.

-spef

Specifies that the mapping mentioned in the name map file will be used during the `read_spef` command. If this option is not specified, the mapping mentioned in name map file will be used while reading simulation data such as `read_fsdb`.

Related Commands

set_rtl_to_gate_name, read_spef, read_fsdb, read_saif, read_qwave

Usage

Reads the RTL to Gate Level netlist name-map file. The name-map file contains a list of `set_rtl_to_gate_name` commands. Note that this command must be called after the `build_design` command and before reading the simulation file (such as, `read_fsdb`) or the SPEF file (such as, `read_spef`), and the global `pa_rtl_sim_on_glpa` must be enabled in case of RTL sim on GLPA flow.

Examples

Example 1: Reads the `design1.saif_map` file and runs the list of `set_rtl_to_gate_name` commands that are present in the file.

```
read_name_map_file design1.saif_map
```

Example 2: Reads the `design1.saif_map` file and runs the `set_rtl_to_gate_name` commands that are listed in the file. The name of the top module `top/design1/u_design` is removed from the values of the `-gate` and `-rtl` options of the `set_rtl_to_gate_name` commands. The commands that do not have the specified top module are ignored.

```
read_name_map_file design1.saif_map -instance top/design1/u_design
```

Example 3: Reads the `design1.saif_map` file and runs the `set_rtl_to_gate_name` commands that are listed in the file. The name of the top module `top/design1/u_design` is removed from the value of the `-gate` option and `top/design1/base/u_design` is removed from the value of the `-rtl` option of the `set_rtl_to_gate_name` commands. The commands that do not have the specified `-instance` and `-rtl_instance` as top modules are ignored.

```
read_name_map_file design1.saif_map -instance top/design1/u_design -
rtl_instance top/design1/base/u_design
```

Example 4: Reads the `design1.saif_map` file and runs
the `set_rtl_to_gate_name` commands that are listed in the file, where the hierarchical separator for the RTL and the gate-level names are `.` and / respectively.

```
read_name_map_file design1.saif_map -rtl_hierarchy_separator "." -
gate_level_hierarchy_separator "/"
```

Example 5: Reads the `design1.saif_map` file and runs the `set_rtl_to_gate_name` commands that are listed in the file, where mapping specified will be used while mapping SPEF constraints.

```
read_name_map_file design1.saif_map -spef
```

---

read_phy_db

read_phy_db - Reads parasitic information from a Calypto-generated parasitic file that has a `.phydb` extension. Note that this command can be run after the `build_design` and before the `prototype_design` commands.

Synopsis

`read_phy_db <`*`phy_file`*`> [-instance_name <`*`inst_name`*`>] [-print_hierarchy]`

<*`phy_file`*>

> Specifies the name of the parasitic file that must be read. The <*`phy_file`*> file must have the `.phydb` extension.

-instance_name <*`inst_name`*>

> Specifies the hierarchical name of the instance for which the parasitic information must be read. This option must be specified if the parasitic file is generated for a bigger design, however, is instead used for a subset of that design.

-print_hierarchy

> Displays the hierarchical structure of the design for which the parasitic file was generated.

Related Commands

read_spef

Usage

The read_phy_db command can be used to read the parasitic files.

Note: This command is under limited production support.

Examples

Example 1: Reads parasitic information from the `design1.phydb` file.

```
build_design design1.v
read_phy_db design1.phydb
prototype_design
```

Example 2: Reads parasitic information for the sub-block A from the `top.phydb` file.

```
build_design A.v
read_phy_db top.phydb -instance top.A
prototype_design
```

read_previous_eco_directives

read_previous_eco_directives — Reads ECO directives from a previous ECO run of PowerPro for the purpose of iterative ECO. Note that this command must be run after the first ECO run and before running any optimization commands in ECO mode.

Synopsis

```
read_previous_eco_directives   -filename <file_name>
```

-filename *<file_name>*

Specifies the name of the ECO directives file dumped in the previous run.

Related Commands

set_powerpro_mode, read_eco_db

Usage

Reads ECO directives from a previous ECO run of PowerPro for the purpose of iterative ECO. Note that optimizations invalidated in the previous ECO run will continue to remain invalid in the current ECO run.

Example

Example: Script to run the first-order and the second-order ECOs.

```
#First Order ECO

set_powerpro_mode eco
read_eco_db <Non ECO run optimization DB>
read_library <library files>
build_design <1st ECO RTL>
prototype_design
insert_obs/insert_stability_logic
write_rtl
write_verify_script
SlecPro verification

#The following is the script to run the second-order ECO. The ECO
directives generated from the first ECO run will be used to invalidate
all the moves in the second ECO run on ECOed-rtl. Note that the ECO
directives file generated in the second ECO run includes directives
from both current and previous ECO runs.

# Second Order ECO

set_powerpro_mode eco
read_eco_db <Non ECO run optimization DB>
read_previous_eco_directives -filename <eco_directives.tcl> #reads
eco_directives.tcl generated by 1st ECO PowerPro run
read_library <library files>
build_design <2nd ECO RTL>
prototype_design
insert_obs/insert_stability_logic # Move invalidated in previous ECO
run will be invalidated
write_rtl
write_verify_script
SlecPro verification
```

read_qwave

read_qwave — Reads the QWAVE file that must be used during simulation. To read a QWAVE file, the installation must have the latest version of the dynamic library file `libddi.so`.

Synopsis

```
read_qwave <filename> -instance_name <hierarchical_instance_name>
    -designfile <filename>
    [-use_name_fallback { on | off }]
    [-complex_name_matching { on | off }]
    [-qwave_name_suffix <suffix_string> ]
    [-begin <begin_time> ]
    [-end <end_time> ]
    [-skip_missing_signals { on | off }]
    [-skip_up_structs { on | off }]
    [-skip_packed_structs { on | off }]
    [-maxbits_size <max_limit>]
    [-print_hierarchy]
    [-zero_delay]
```

`<filename>`

Specifies the name of the QWAVE file.

-instance_name *`<hierarchical_instance_name>`*

Specifies the hierarchical name of the top module of the design, as specified in the QWAVE file.

-designfile *`<filename>`*

Specifies the absolute path to the `design.bin` file. This option is required.

-use_name_fallback { on | off }

When set to `on`, if the signal name obtained from the QWAVE file does not match any signal names in the design, the signal name obtained from the QWAVE file is modified to remove the escape characters and indices before attempting a match again. Default is on.

-complex_name_matching { on | off }

When set to `on`, PowerPro matches such as the generate-block names from the QWAVE file. Default is `on`.

Note: While set to `on`, it leads to better switching activity assertion, and as a result, to increased run time.

-qwave_name_suffix *`<suffix_string>`*

Specifies that the names in the QWAVE file contain a suffix *`<suffix_string>`* that must be removed before matching a signal or instance name obtained from the QWAVE file to a signal or instance name in the design.

-begin *`<begin_time>`*

Specifies the start time of the simulation window. If the time unit is not specified, PowerPro automatically infers it from the QWAVE file. The supported time units are `s, ms, us, ns, ps, fs, as, za and ys.`

-end *`<end_time>`*

Specifies the end time of the simulation window. If the time unit is not specified, PowerPro automatically infers it from the QWAVE file. The supported time units are `s, ms, us, ns, ps, fs, as, za and ys.`

-skip_missing_signals { on | off }

When set to `off`, adds another pass over the design to increase switching activity assertion. Default is `on`.

-skip_up_structs { on | off }

When set to `on`, unpacked structure constructs are ignored. Default is `on`.

-skip_packed_structs { on | off }

When set to `on`, packed structure constructs are ignored. Default is `on`.

-maxbits_size <*`max_limit`*>

Specifies the maxbits dump size limit used during the qwave.db generation. If the maxbits size is not specified, PowerPro automatically defaults to `4096`.

-print_hierarchy

Displays the hierarchical structure of the design for which the QWAVE file was created. To use the `print_hierarchy` option, the `filename` <*`filename`*> and `designfile` <*`filename`*> options must be specified. If this option is specified, all other options are ignored.

-zero_delay

Specifies that the QWAVE file contains no delay information, and power computation will happen using Zero Delay Power Computation Engine. When this option is not specified, PowerPro will assume that QWAVE file has delay information and will use Delay Aware Power Computation Engine. Note that this option will only have effect on GLPA flow with the `-pa` option.

Related Commands

read_fsdb

Usage

The **read_qwave** command specifies the QWAVE file that must be used during sequential optimization. Note that the switching activity is automatically inferred from the QWAVE file.

Multiple `read_qwave` commands are not allowed in the same session.

Examples

Example 1: Reads the QWAVE file `qwave.db`. During simulation, only those signals that fall within the scope of the instance `testbench.dut` are considered. Note that in the following example, the instance name `testbench` is the name of the testbench module, and `dut` is the name of the instance corresponding to the top module of the design within the testbench.

```
read_qwave qwave.db -designfile design.bin -instance_name testbench.dut
```

Example 2: Reads the QWAVE file `qwave.db` and annotates the signals that fall within the scope of the instance `testbench.dut`, within the simulation window `1000000000us` and `1010000000us`. In this example, since the time unit is not specified, it is automatically inferred from the QWAVE file.

```
read_qwave -instance_name testbench.dut qwave.db -designfile design.bin
-begin 1000000000 -end 1010000000
```

Example 3: Reads the QWAVE file `qwave.db` and annotates the signals that fall within the scope of the instance `testbench.dut`, within the simulation window `1000000000000` to `4000000000000` on the qwave scale `1fs`. In this example, while the QWAVE scale is in `fs`, the user specifies the time unit in the `read_qwave` command as `ms`. In such cases, PowerPro automatically converts the time specified in the `read_qwave` command to use the time scale specified in the QWAVE file.

```
read_qwave -instance_name testbench.dut qwave.db -designfile design.bin
-begin 1ms -end 4ms
```

read_saif

read_saif — Reads switching activity information from a SAIF file(s) to enable accurate power analysis and optimization.

Synopsis

```
read_saif [ -instance_name <instance> ]
    [ -target_instance <target_instance> ]
    [ -scale <unit_scale_value> ] [ -unit_base <unit_type> ]
    [ -function_type <func_type>] [ -using_hier_name ]
    [ -saif_instance_name_suffix <suffix>] [ -weight <weight> ]
    [ -incremental ] [ -name_divider <hierarchy_separator> ]
    [ -case_insensitive ]
    { <saif-filename> | -weight <weight> <saif-filename>... }
    [ -print_hierarchy ]
    [-annotate_tech_cell ]
```

-instance_name *<instance>*

Specifies the hierarchical name of the instance from the SAIF file that must be used as the starting point for parsing switching activity information. If no such instance is provided, the hierarchy two levels below, as specified in the SAIF file, is used.

-target_instance *<target_instance>*

Specifies the name of the instance in the current design on which switching activity information needs to be annotated from the SAIF file. If no such instance is provided, then the top level hierarchy is assumed to be the target.

-scale *<unit_scale_value>*

Specifies the scaling factor for the base synthesis time unit from the SAIF file. Legal values include positive integers greater than `1`.

For example, specifying `-scale 10 -unit_base ps` sets the desired time unit as `10 ps`. If a scaling factor is not provided, the scaling factor is determined from the SAIF file.

Note: This option requires the `-unit_base` option.

-unit_base *<unit_type>*

Specifies the base synthesis time unit. Legal values are `fs`, `ps`, `ns`, `us`, `ms`, and `s`. Default is `ns`.

-function_type *<func_type>*

If multiple SAIF files are specified, `read_saif` performs a weighted average on the switching activity data - <prob,td>. To override this behavior, use this option. Legal values are `max_td`, `max_prob`, and `max`.

Specify `max_td` to force `read_saif` to use the maximum `td` value for an object across all SAIF files; accordingly, the associated `prob` value would be used.

Specify `max_prob` to force `read_saif` to use the maximum `prob` value for an object across all SAIF files; accordingly, the associated `td` value would be used.

Specify `max` to force `read_saif` to use the maximum `prob` and maximum `td` value for all objects across all SAIF files.

-using_hier_name

Specifies that the SAIF file being read is a flat SAIF file. In flat SAIF files, instance names are written as `(INSTANCE testbench.dut.I1.I2 ... )`, instead of using the general hierarchical structure.

-saif_instance_name_suffix *<suffix_string>*

Specifies that the suffix, added during SAIF file generation, be removed from the instance name before annotating the instance with switching activity information. This helps in increasing the percentage assertion of sequential elements.

-weight *<weight>*

Specifies the value by which switching activity data in the SAIF file must be scaled before annotating on the switching activity data. Legal values are float values >0 and <=1.

The sum of weights must be equal to `1`, except when the `-incremental` option is specified. In case this option is not specified, the SAIF file is assigned a weight of `1`.

-incremental

Specifies that additional switching activity from a different SAIF file needs to be annotated with a weight on a previously annotated design net. In this case, the weight of the previously annotation would be considered as: `1 – <weight_specified>`.

Consider the following example:

read_saif saif_file1.saif read_saif -incremental -weight 0.25 saif_file2.saif

Effectively, this assigns a weight of .75 to saif_file1.saif and .25 to saif_file2.saif. This is equivalent to specifying the following command:

read_saif -weight 0.75 saif_file1.saif -weight 0.25 saif_file2.saif

-name_divider *<hierarchy_separator>*

Specifies the hierarchy separator. By default the `"."` is used to demarcate hierarchy. Note that this option is honored only for hierarchical names specified with `-instance_name`.

-case_insensitive

Specifies that the case of the signal/instance name be ignored when matching names for annotation. This helps in increasing the percentage assertion of signals and sequential elements.

<saif-filename> | -weight *<weight>* *<saif-filename>*...

<saif-filename> specifies the name of the SAIF file. `read_saif` also accepts files in compressed formats as an argument. Supported formats include: .Z, .tar.gz and .gz.

Multiple SAIF files can be also be specified; however, a weight must be associated with each file. Refer to the `-weight` option for more information. The *Examples* section provides more information on reading-in multiple SAIF files.

-print_hierarchy

Displays the hierarchical structure of the design for which the SAIF file was created. This option expects the name of the SAIF file to be specified. When this option is specified, all other options are ignored.

-annotate_tech_cell

Annotates the Switching Activity of the tech cell from SAIF file in RTL power analysis flow. This option is ON by default in Gate level power analysis flow.

Related Commands

read_fsdb

Usage

The **read_saif** command annotates switching activity information from one or more SAIF files. Each SAIF file has an associated weight. This weight is used to scale switching activity values before annotating the design.

Note: When the -print_hierarchy option is specified, the read_saif command only prints the hierarchy of the SAIF file; it does not read in the SAIF file. To read the SAIF file, specify the read_saif command without the -print_hierarchy option.

Examples

In examples 1, 2 and 3, it is assumed that I1 is an instance (of module M1) in the top module of the design and I2 is an instance (of module M2) inside I1. Another assumption is, SAIF file saiffile.saif is generated by instantiating the entire design as dut in the module testbench.

Example 1: Reads the instance testbench.dut.I1.I2 and annotates switching activity information on the target instance M2 (as set by the build_design command).

```
build_design  -top M2 test.v
read_saif  -instance_name testbench.dut.I1.I2 saiffile.saif
```

Likewise, if the file saiffile.saif was compressed as saiffile.saif.Z, the command would be:

```
build_design  -top M2 test.v
read_saif  -instance_name testbench.dut.I1.I2 saiffile.saif.Z
```

Example 2: Starting from the instance testbench.dut.I1.I2, the read_saif command parses nets and the corresponding information about the switching activity, and annotates the information in the design instance I1.I2.

```
build_design test.v
read_saif  -target_instance I1.I2  -instance_name testbench.dut.I1.I2
saiffile.saif
```

Example 3: Extending the same example used in *Example 2*, if the scope is changed to apply to the top module M1, read_saif will start processing processing switching activity information from testbench.dut.I1.I2 and only populate switching activity associated with the instance I2.

```
build_design  -top M1 test.v
read_saif  -target_instance I2  -instance_name testbench.dut.I1.I2
saiffile.saif
```

Example 4: This example illustrates behavior when the scaling of switching activity varies and some nets exist only in one SAIF file and not the other.

Given the following assumptions:

```
saif_1.saif has the nets n1, n2, n3
saif_2.saif has nets n2, n3
saif_3.saif has nets n1, n3, n4
```

Running the following command:

```
read_saif  -weight 0.1 saif_1.saif  -weight 0.4 saif_2.saif  -weight
0.5 saif_3.saif
```

will annotate switching activity information as follows:

```
SA(n1) = (0.1*saif_1 + 0.5*saif_3) / (0.1 + 0.5)
SA(n2) = (0.1*saif_1 + 0.4*saif_2) / (0.1 + 0.4)
SA(n3) = (0.1*saif_1 + 0.4*saif_2 + 0.5*saif_3) / (0.1 + 0.4 + 0.5)
SA(n4) = (0.5*saif_3) / (0.5) = saif_3
```

Example 5: This example illustrates how switching activity information can be evenly distributed from 4 different SAIF files to a design net. Specify the following commands to achieve equivalent weightage for the probability values P1, P2, P3 and P4:

```
read_saif  saif1.saif                           // Command1
read_saif  -incremental saif2.saif   -weight 0.5       // Command2
read_saif  -incremental saif3.saif   -weight 1/3       // Command3
read_saif  -incremental saif4.saif   -weight 1/4       // Command4
```

By running the above mentioned commands, the following probability values will be obtained:

- After running Command1, the edge will have a probability of `P1`.
- After running Command2, the edge will have a probability value of `(P1/2 + P2/2 )`.
- After running Command3, the edge will have a probability value of `2/3 * ( P1/2 + P2/2 ) + P3/3`, which is the same as `P1/3 + P2/3 + P3/3`.
- After running Command4, the edge will have a probability value of `3/4 * ( P1/3 + P2/3 + P3/3 ) + P4/4`, which is the same as `P1/4+ P2/4 + P3/4 + P4/4`.

Example 6: This example shows the usage of the `-using_hier_name` option. Assuming that in a flat SAIF file, the names of INSTANCE are printed as:

```
(INSTANCE testbench.dut)
(INSTANCE testbench.dut.A)
(INSTANCE testbench.dut.A.B)
(INSTANCE testbench.dut.A.B.C)
```

Here, the value for `-instance_name` should be such that if this value is stripped from the hierarchical name in the SAIF file, the remaining part continues to be a valid hierarchical name. For example, in the above SAIF snippet, `A`, `A.B`, `A.B.C` are valid hierarchical names in the design so `testbench.dut` can be provided as a value to `-instance_name` as follows:

```
read_saif  -instance_name testbench.dut  -using_hier_name saiffile.saif
```

Example 7: Searches the SAIF file `saiffile.saif` for the instance name `inst` instead of `inst_reg`. This is assuming the SAIF file has sequential element names with the suffix `reg`.

```
read_saif  -saif_instance_name_suffix "_reg" saiffile.saif
```

Example 8: By default, `read_saif` does a weighted average on the switching activity `<prob,td>` of an object obtained from multiple SAIF files. This example illustrates how the `-function_type max_td` option can be used to override this default behavior. Instead, `read_saif` considers the maximum td value for that object across all SAIF files. Accordingly, the probability value associated with the maximum td is considered.

Assuming the signal `net_1` has SA data `<0.9, 0.5>` in the file `SAIF_1.saif` and SA data `<0.1,0.9>` in the file `SAIF_2.saif`, the following command will force PowerPro to use the SA data `<0.1, 0.9>` for signal `net_1`.

```
read_saif SAIF_1.saif SAIF_2.saif  -function_type max_td
```

Example 9: Prints the structure of the SAIF file SAIF_1.saif. Switching activity information is not read. Sample results are also shown below.

```
powerpro> read_saif  -print_hierarchy SAIF_1.saif

testbench
testbench.dut
testbench.read_input_file
testbench.simulate
testbench.dut.cs_reg_reg
testbench.dut.mem_inst
testbench.dut.raddr_reg_reg
testbench.dut.rw_reg_reg
testbench.dut.waddr_reg_reg
testbench.dut.wdata_reg_reg
testbench.dut.mem_inst.calypto_mem_inst
```

Example 10: Starting from the instance `{testbench/dut/f1/LinkGen\[0\].MasterLink/LinkLayer}`, the `read_saif` command parses nets and the corresponding information about the switching activity, and annotates the information in the design instance `{f1.\LinkGen[0].MasterLink.LinkLayer}`.

```
build_design test.v
read_saif -instance_name
{testbench/dut/f1/LinkGen\[0\].MasterLink.LinkLayer} -target_instance
{f1.\LinkGen[0].MasterLink.LinkLayer } $testdir/gold.saif -name_divider
"/"
```

Example 11: Annotates the switching activity of the tech cell from the saiffile.saif. This option is on by default in Gate level power analysis flow.

```
build_design test.v
read_saif saiffile.saif -instance_name testbench.dut -
annotate_tech_cell
```

read_sdc

read_sdc — Reads in a Synthesis Design Constraints (SDC) file. SDC version 2.1 and earlier are supported. Note that this command can only be called after `build_design`.

Synopsis

```
read_sdc  <sdc_file>
```

<sdc_file>

Specifies the name(s) of the SDC file(s) to be read in.

Related Commands

all_clocks, all_inputs, all_outputs, all_registers, create_generated_clock, get_clocks, get_nets, get_ports, set_case_analysis, set_logic_one, set_logic_zero

Usage

The **read_sdc** command reads in SDC files. When multiple SDC files are read in and the SDC files contain calls to the SDC commands `current_instance` and `set_hierarchy_separator`, the scope of these commands will be limited to the files read within a single `read_sdc` call.

Examples

Example 1: Reads -in one SDC file at a time.

```
read_sdc test1.sdc
read_sdc test2.sdc
```

In this example, the scope of the `current_instance` and `set_hierarchy_separator` commands will be limited to the SDC file read within the `read_sdc` call. So, the `current_instance` and `set_hierarchy_separator` commands for the first `read_sdc` call will apply only to `test1.sdc` while the second `read_sdc` command will apply only to `test2.sdc`.

Example 2: Reads -in multiple SDC files in a single `read_sdc` call.

```
read_sdc test1.sdc test2.sdc
```

In this example, the SDC commands `current_instance` and `set_hierarchy_separator` specified in `test1.sdc` will apply to both `test1.sdc` and `test2.sdc`.

read_spef

read_spef — Reads parasitic information from a Standard Parasitic Extraction Format (SPEF) file. Note that this command can be run only after the `build_design` and before the `prototype_design` commands.

Note: If the SPEF is read, the `calypto.phydb` file will be automatically created in the work directory and can be used by the `read_phy_db` command.

Synopsis

```
read_spef <spef_file> [ -instance_name <spef_inst_name> ]
    [ -target_instance <inst_name> ] [ -print_hierarchy ]
```

`<spef_file>`

Specifies the name of the SPEF file that must be read. The *<spef_file>* must have the `.spef` extension, or it must be a compressed file in the `.tar.gz` or `.gz` format.

-instance_name *`<spef_inst_name>`*

Specifies the hierarchical name of the instance as given in the SPEF file for which the parasitic information must be read. This option must be specified if the parasitic file is generated for a bigger design, however, is instead used for a subset of that design.

-target_instance *`<inst_name>`*

Specifies the name of the instance in the current design for which the SPEF file must be read. If the option is not specified, the top level hierarchy is assumed to be the target instance.

-print_hierarchy

Displays the hierarchical structure of the design for which the SPEF file was generated. This option expects the name of the SPEF file. If this option has been specified, all other options must be ignored.

Related Commands

set_load

Usage

The **read_spef** command reads SPEF files. Files in compressed formats are supported as well.

Note: When the `-print_hierarchy` option is specified, the `read_spef` command only prints the hierarchy of the SPEF file; it does not read in the SPEF file. To read parasitic data from the SPEF file, specify the `read_spef` command without the `-print_hierarchy` option.

Example

Example 1: Reads parasitic data from the file `spef_file.spef`

```
read_spef spef_file.spef
```

Example 2: Reads parasitic data from the compressed SPEF file `spef_file.spef.tar.gz`.

```
read_spef spef_file.spef.tar.gz
```

Example 3: Prints the structure of the compressed SPEF file `spef_file.spef.tar.gz`. Parasitic data is not read. Sample results are shown below.

```
powerpro> read_spef  -print_hierarchy spef_file.spef.tar.gz

RC_CG_SHARED_HIER_L1_INST
correlate_rows
count_inst
decode_row
ecc_inst
two_tap_row_a
two_tap_row_b
count_inst.RC_CG_HIER_INST0
decode_row.RC_CG_HIER_INST1
decode_row.RC_CG_HIER_INST2
ecc_inst.RC_CG_HIER_INST3
two_tap_row_a.RC_CG_HIER_INST4
two_tap_row_b.RC_CG_HIER_INST6
```

Example 4: Starting from the instance `{L1/L2/L3}`, the read_spef command parses nets in the spef instance {L1/L2/L3} and annotates the information on design's top instance.

```
powerpro> read_spef -instance_name L1/L2/L3 design.spef
```

Example 5: read_spef command parses nets in the spef and annotates information on design instance {`L1/L2/L3}`.

```
powerpro> read_spef -target_instance L1.L2.L3 design.spef
```

But if SPEF is not read on top hierarchy, proto will error out.

Example 6: Starting from the instance `{L1/L2/L3}`, the read_spef command parses nets in the spef instance `{L1/L2/L3}` and annotates the information design's on instance `L1.L2.L3`.

```
powerpro> read_spef -target_instance L1.L2.L3 -instance_name L1/L2/L3 
design.spef
powerpro> read_spef -target_instance L1.L2 -instance_name L1/L2 
design1.spef
powerpro> read_spef -target_instance L1 -instance_name L1 design2.spef
powerpro> read_spef design4.spef

But if SPEF is not read on top hierarchy, proto will error out.
```

If there are different SPEF files for different instances in design, then it is mandatory to read SPEF file for innermost hierarchy.

Example 7:

```
powerpro> read_spef -target_instance L1.L2 -instance_name L1/L2 
design1.spef
powerpro> read_spef -target_instance L1.L2.L3 -instance_name L1/L2/L3 
design.spef
```

This set of commands will result in error.

[PHY-SDFIE] SPEF information for instance 'L3' already exists.

Please try 'help PHY-SDFIE' for more detailed

NOTE: This error occurs only in RTL flow not in gate level flow.

read_vcd

Unsupported w.e.f. PowerPro 10.1. Calypto does not recommend the use of this command in the flow.

read_vcd — Specifies the VCD file to be used during simulation.

Synopsis

```
read_vcd { -vcd <vcd_file_name> | <filename> }
    [ -instance_name <hierarchical_instance_name> ]
    [ -use_name_fallback ] [ -vcd_name_suffix <suffix_string> ]
```

-vcd *<vcd_file_name>* | *<filename>*

Specifies the name of the vcd file.

-instance_name *<hierarchical_instance_name>*

Specifies the hierarchical name of the top module of design, as specified in the VCD file.

-use_name_fallback

Specifies that the name of the signal in the VCD file be modified for escape characters and indices before attempting a rematch of the name when a match is not found.

-vcd_name_suffix `<suffix_string>`

Specifies that PowerPro remove the suffix `<suffix_string>` before matching the instance or signal name in the VCD file.

Related Commands

read_fsdb, read_saif

Usage

The **read_vcd** command specifies the VCD file to be used during simulation. Note that reading in a VCD file can cause an increase in runtime and memory and might also result in low switching activity assertions. It is recommended that the `read_fsdb` command be used, instead. If using the `read_vcd` command, make sure that the `read_saif` command is called prior to calling the `read_vcd` command. This is required to gather switching activity information for the design objects.

Examples

Example 1: Reads in the VCD file `design.vcd`. During simulation only signals within the scope of the instance `testbench.dut` are considered, where `testbench` is the name of the module and `dut` is the name of the instance.

```
read_saif sample.saif
read_vcd  -vcd design.vcd -instance_name testbench.dut
```

read_veloce_dataset

read_veloce_dataset — Reads the STW file that is used to compute power. To read an STW file, the installation must have the latest version of the dynamic library file `libVelWave.so`. The command must be run before the `prototype_design` command.

Synopsis

```
read_veloce_dataset <filename>
    -instance_name <hierarchical_instance_name>
    [-begin <begin_time>] [-end <end_time>] [-live]
```

<filename>

Specifies the name of the STW file.

-instance_name *<hierarchical_instance_name>*

Specifies the hierarchical name of the top module of the design, as specified in the STW file. The hierarchical name must use the correct hierarchy delimiter so that the name can be found in the STW file. For VHDL and Mixed language designs, specify the hierarchy delimiter '/'.

-begin *<begin_time>*

Specifies the start time of the emulation window. If the time unit is not specified, PowerPro automatically infers it from the STW file. The supported time units are `s`, `ms`, `us`, `ns`, `ps`, `fs`, `as`, `za`, and `ys`.

-end *<end_time>*

Specifies the end time of the emulation window. If the time unit is not specified, PowerPro automatically infers it from the STW file. The supported time units are `s`, `ms`, `us`, `ns`, `ps`, `fs`, `as`, `za`, and `ys`.

-live

Specifies that the emulation is currently running and the streamed data is being simultaneously consumed by PowerPro.

Related Commands

read_fsdb, read_qwave

Usage

The **read_veloce_dataset** command reads the STW file that must be used for power computation. Note that the switching activity is automatically inferred from the STW file.

Note: This command is under limited production support.

Examples

Example 1: Reads the STW file `mytrace.stw`. During emulation, only those signals that fall within the scope of the instance `testbench.dut` are considered. Note that in the following example, the instance name `testbench` is the name of the testbench module, and `dut` is the name of the instance corresponding to the top module of the design within the testbench.

```
read_veloce_dataset .../../mytrace.stw -instance_name testbench.dut
```

Example 2: Reads the STW file `mytrace.stw` and annotates those signals that fall within the scope of the instance `testbench.dut`, within the simulation window `1000000000us` and `1010000000us`. In this example, since the time unit is not specified, it is automatically inferred from the STW file.

```
read_veloce_dataset .../../mytrace.stw -instance_name testbench.dut -
begin 1000000000 -end 1010000000
```

Example 3: Reads the STW file `mytrace.stw` and annotates those signals that fall within the scope of the instance `testbench.dut`. Note that the emulation is currently running and the streamed data is being simultaneously consumed by PowerPro.

```
read_veloce_dataset -live .../../mytrace.stw -instance_name
testbench.dut
```

rebind_flop_macro

rebind_flop_macro — Binds a flop macro to another flop macro and inserts an enable at the specified position.

Synopsis

```
rebind_flop_macro  -from <macro_to_bind>  -to <macro_with_enable>
    -enable_pos <position>
```

-from *<macro_to_bind>*

Specifies the name of the flop macro which is to be bound to the other macro.

-to *<macro_with_enable>*

Specifies the name of flop macro which has the enable pin. During `write_rtl`, PowerPro replaces *<macro_to_bind>* with *<macro_with_enable>*, if a move is inserted for this flop.

-enable_pos *<position>*

Specifies the position at which enable signal is to be inserted, where *<position>* is an integer value and count starts from `0`.

Related Commands

config_write_rtl

Usage

The `rebind_flop_macro` command rebinds a flop macro to another flop macro and inserts an enable at the specified position. The original flop macro is replaced only when a move is inserted on the flop, else it is preserved as it is.

Note: For this command to be honored, the `config_write_rtl -preprocess_verilog` option must be set to true.

Examples

Example 1: Replaces the flop macro `MSFF` with the flop macro `MSFF_EN` and inserts the enable signal at position 3 when a move is inserted on the flop `sff`.

Consider the following example.

```
// Original rtl:

`define MSFF_EN(q,i,clk,en)    \
       always @ (posedge clk) \
         if ( en ) \
             q<i;

`define MSFF(q,i,clk)   \
       always @ (posedge clk)    \
          q<i;

module flop (sff,sel,clk_1)
          `MSFF(sff,sel,clk_1) ;
endmodule
```

To insert the enable at position 3 and replace the flop macro `MSFF` with `MSFF_EN`, specify the following command:

```
rebind_flop_macro  -from MSFF  -to MSFF_EN  -enable_pos 3
```

The resulting patched RTL is shown below:

```
`define MSFF_EN(q,i,clk,en)   \
      always @ (posedge clk) \
         if ( en )     \
              q<i;

`define MSFF(q,i,clk)   \
      always @ (posedge clk)   \
         q<i;

module flop (sff,sel,clk_1)
     //`MSFF(sff,sel,clk_1) ;
      - - PowerPro -CG
        `MSFF_EN(sff,sel,clk_1,ppro_en) ;
endmodule
```

report_area

Deprecated w.e.f. PowerPro 10.4.

Reports area usage of the design. The information is presented as a table, in plain text, XML or CSV format.

Synopsis

```
report_area [ -module <module_name> |  -inst <instance_name> ]
    [ -hier] [ -no_wrap] [ -text |  -xml |  -csv ]
    [ -csv_delimiter <delimiter>] [ -filename <filename>]
```

-module *<module_name>* | -inst *<instance_name>*

> Specifies the module or instance for which area usage is to be reported. By default, area usage is reported for the entire design.

-hier

> Provides area usage for modules in the hierarchy as well. By default, area usage is reported for the top module. In case a module or instance is specified, area usage is reported for the specific module/instance only.

-no_wrap

> By default, information is printed in fixed -width columns. If information for a field exceeds the column width, information continues on the next line under the same column. This option prevents information in columns from splitting across rows, making it easier to extract information from the report.

-text | -xml | -csv

> Specifies the format of the report file to be generated. The information is reported in table format, with a choice of saving the report in text, XML and CSV format. By default, the report is generated as a plain text file.
>
> Note: For CSV files, the semicolon(`;`) is used as the default delimiter. To specify another delimiter, use the `-csv_delimiter` option.

-csv_delimiter *<delimiter>*

> Specifies the delimiter for the CSV report. By default, the semicolon (;) is used as a delimiter.

-filename *<file_name>*

> Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

Usage

The **report_area** command generates and outputs a report on the area usage of the design. The information is presented in a table, in plain text, XML or CSV format.

The different columns in the table are listed below:

- Instance. Displays the name of the instance, if applicable. Note: For the top level module, this column is always blank.

- Module/Cell. Displays the name of the module or leaf -level cell linked to the instance. This column displays the design name for the top level design.
- Cell Area. Displays the sum of the areas of all instances of the leaf -level cells hierarchically contained in the instance or module.
- Net Area. Displays the sum of the areas of all nets hierarchically contained entirely within the instance or module.
- Total Area. Displays the sum of the cell and net areas.
- Instance Count. Displays the number of instances of modules and leaf -level cells hierarchically contained in the instance or module.
- Reg Count. Displays the number of sequential elements hierarchically contained in the instance or module.
- Comb Area. Displays the sum of the areas of all instances of combinational leaf -level cells hierarchically contained in the instance or module.
- Seq Area. Displays the sum of the areas of all instances for sequential leaf -level cells hierarchically contained in the instance or module.

Each entry in the table corresponds to a top level module in the design. Normally, there is just one top level module, so invoking `report_area` without arguments results in a table (in text format) with just one entry being sent to the standard output. (See the first entry in the Examples section, below.)

Examples

Example 1: Reports information on the total area usage of the design.

```
powerpro> report_area
Report: area
---------------------------------------------------------------------------------------------------
Instance     Module/Cell    Cell Area  Net Area  Total Area  Inst Count  Reg Count  Comb Area  Seq Area
---------------------------------------------------------------------------------------------------
             calypto_top    197934     0         197934      2127        170        189620     8313.6
```

Example 2: Reports information on the area usage of the hierarchical instance `g_hier_level_0`, including a breakdown of the hierarchical instances contained within `g_hier_level_0`.

```
powerpro> report_area -inst inner_level_1 -hier
Report: area
-------------------------------------------------------------------------------------------------------------
Instance                Module/Cell      Cell Area  Net Area  Total Area  Inst Count  Reg Count Comb Area  Seq Area
-------------------------------------------------------------------------------------------------------------
inner_level_1           MOD_INNER_LEVEL_1
                                         123        0         123         17          0         0          0
inner_level_1.inner_level_2
                        MOD_INNER_LEVEL_2
                                         68         0         68          11          0         0          0
```

report_clocks

report_clocks — Reports information about clocks in the design. The report includes information about domain name, average toggle density along with width of clock root, clock period, type of clock, actual clock roots involved and whether or not the reset port has been set for the clock.

Synopsis

```
report_clocks [ -clock_roots <clock_roots_name>]
    [ -filename <filename>] [ -text |  -xml |  -csv ]
    [ -csv_delimiter <csv_delimiter>] [ -no_wrap]
```

-clock_roots *`<clock_roots_name>`*

Specifies the clock roots for which report is to be created. By default, report is created for all the clock roots in design.

-filename *`<filename>`*

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

-text | -xml | -csv

Specifies the format of the report file to be generated. The information is reported in table format, with a choice of saving the report in text, XML and CSV format. By default, the report is generated as a plain text file.

Note: For CSV files, the semicolon(`;`) is used as the default delimiter. To specify another delimiter, use the `-csv_delimiter` option.

-csv_delimiter *`<csv_delimiter>`*

Specifies the delimiter for the CSV report. By default, the semicolon (;) is used as a delimiter.

-no_wrap

By default, information is printed in fixed -width columns. If information for a field exceeds the column width, information continues on the next line under the same column. This option prevents information in columns from splitting across rows, making it easier to extract information from the report.

Related Commands

create_clock, set_powerpro_reset

Usage

The **report_clocks** command reports information about clocks in the design. This includes information about both user-specified clocks, as well as PowerPro inserted clocks. The report includes information about the clock domain name, clock root port, type of clock (User/PowerPro created), clock period and toggle rate for each clock.

Tip! The output of this command can be used to identify missing PowerPro resets. When running `insert_stability_logic`, if a reset is missing for an internal clock which was created during complex/constant clock handling, an error will be generated. The output of this command can prove useful in identifying the missing reset.

Examples

Example 1: Reports information for all the clocks that have the clock root `'clk'`. The report is written to a text file named `clocks_report.rpt`.

```
report_clocks  -clock_roots clk  -filename clocks_report.rpt
```

Example 2: Reports information for all the clocks that have the clock root '`clk2`' or `'xgmii_rxclk'`.

```
report_clocks  -clock_roots { clk2 xgmii_rxclk }
```

report_clock_gated_memories

Deprecated w.e.f. PowerPro 10.4.

report_clock_gated_memories — Reports all memories in design that have been clock gated by PowerPro. Note that this command can only be called after `insert_mem_clock_gating`.

Synopsis

```
report_clock_gated_memories [ -no_wrap] [ -text |  -xml |  -csv ]
    [ -csv_delimiter <delimiter> ] [ -filename <filename> ]
```

-no_wrap

By default, information is printed in fixed-width columns. If information for a field exceeds the column width, information continues on the next line under the same column. This option prevents information in columns from splitting across rows, making it easier to extract information from the report.

-text | -xml | -csv

Specifies the format of the report file to be generated. The information is reported in table format, with a choice of saving the report in text, XML and CSV format. By default, the report is generated as a plain text file.

Note: For CSV files, the semicolon(`;`) is used as the default delimiter. To specify another delimiter, use the `-csv_delimiter` option.

-csv_delimiter *<delimiter>*

Specifies the delimiter for the CSV report. By default, the semicolon (;) is used as a delimiter.

-filename *<filename>*

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

Related Commands

insert_mem_clock_gating

Usage

The **report_clock_gated_memories** command reports all those memories in the design that have been clock gated by PowerPro. The report also includes information about the size of the memory, the clock ports gated, power saved and area changed after clock gating.

Examples

Example 1: Generates a report containing information about all memories that have been clock -gated by PowerPro.

```
report_clock_gated_memories
```

report_clock_gating

report_clock_gating — Reports the summary information related to enhanced clock gating that is performed on registers in the Power Analysis Flow.

This command can be called only after the `prototype_design` command.

Note that this command reports information based on the `opt_cg_min_size` variable specified by the user. The information is related to enhanced clock gating performed only in PowerPro. Therefore, the clock gating performed in any other way, for example, clock gating already implemented in the RTL code, will not appear in the report.

Synopsis

```
report_clock_gating [ -gated <true | false>|
                      -ungated <true | false> |
                      -gating_elements <true | false> ]
```

-gated <*true* | *false*>

Reports the number of registers on which clock gating has been performed in PowerPro. The registers on which the clock gating has been performed in the user RTL are not reported. Possible values are `true` and `false`. Default value is `true`.

-ungated <*true* | *false*>

Reports the number of registers on which clock gating has not been performed in PowerPro. The report includes the registers that are already clock-gated in user RTL. Possible values are `true` and `false`. Default value is `true`.

-gating_elements <*true* | *false*>

Reports the number of clock gating cells used in PowerPro. The report does not include the cells that are already instantiated for clock gating in RTL. Possible values are `true` and `false`. Default value is `true`.

Related Commands

prototype_design, report_power, insert_observability_logic, insert_stability_logic

Usage

The `report_clock_gating` command reports clock gating opportunities found on registers during the Power Analysis flow.

Examples

Example 1: Generates the report displaying the summary information related to enhanced clock gating.

```
powerpro> report_clock_gating
Number of Clock gating elements : 1
Number of Gated Registers : 8 (47.06%)
Number of Ungated Registers : 9 (52.94%)
Total Number of Registers : 17 (100%)
```

Example 2: Generates the report displaying the summary information related to enhanced clock gating if the `ungated` option is set to `false`.

```
powerpro> report_clock_gating -ungated false
Number of Clock gating elements : 1
Number of Gated Registers : 8 (47.06%)
Total Number of Registers : 17 (100%)
```

Example 3: Generates the report displaying the summary information related to enhanced clock gating if the `gating_elements` option is set to `false`.

```
powerpro> report_clock_gating -gating_elements false
Number of Gated Registers : 8 (47.06%)
Number of Ungated Registers : 9 (52.94%)
```

Total Number of Registers : 17 (100%)

report_clock_tree_power

report_clock_tree_power — Reports clock tree power for all clock roots in the design.

Synopsis

```
report_clock_tree_power [ -clock_root <clock_port_list> ]
    [ -summary | -detailed ] [ -no_wrap] [ -text | -xml | -csv ]
    [ -filename <filename> ]
```

-clock_root `<clock_port_list>`

> Reports information specific to the clock roots listed by `<clock_port_list>`.

-summary

> Reports a summary of clock information for all clock roots.

-detailed

> Reports the detailed clock information for all clock roots, as well as current global environment settings.

-no_wrap

> By default, information is printed in fixed-width columns. If information for a field exceeds the column width, information continues on the next line under the same column. This option prevents information in columns from splitting across rows, making it easier to extract information from the report.

-text | -xml | -csv

> Specifies the format of the report file to be generated. The information is reported in table format, with a choice of saving the report in text, XML and CSV format. By default, the report is generated as a plain text file.

-filename `<filename>`

> Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

Related Commands

config_clock_tree, report_power

Usage

The `report_clock_tree_power` command reports clock tree power for all the clock roots in the design. This command can be used to generate a summary report showing clock information in tabular format or generate a detailed report for one or more clock roots along with environment information.

Examples

Example 1: Generates a report containing clock tree power information for all clocks in the design.

```
read_library techlib.lib
build_design input.v
create_clock -name clk -period 20 clk
config_clock_tree -buffer BUF1 -fanout 2
prototype_design
report_clock_tree_power
report_power
```

report_data_gating

Deprecated w.e.f. PowerPro 10.4. Instead of this command, use `report_format -metric "Data Gating"`.

report_data_gating — Reports data gating opportunities found on the operators in the design, along with power changes, efficiency changes, and area changes for these operators. Note that this command can be run only after the `insert_observability_logic` command if the global `opt_enable_data_gating` is enabled and the `generate_guidance` command has been run in the flow.

Synopsis

```
report_data_gating [ -filename <filename> ] [ -no_wrap ]
    [ -text | -csv ] [ -csv_delimiter <csv_delimiter> ]
```

-filename *<filename>*

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

-no_wrap

By default, information is printed in fixed -width columns. If information for a field exceeds the column width, information continues on the next line under the same column. This option prevents information in columns from splitting across rows, making it easier to extract information from the report.

-text | -csv

Specifies the format of the report file to be generated. The information is reported in table format, with a choice of saving the report in text and CSV format. By default, the report is generated as a plain text file.

Note: For CSV files, the semicolon(;) is used as the default delimiter. To specify another delimiter, use the `-csv_delimiter` option.

-csv_delimiter *<csv_delimiter>*

Specifies the delimiter for the CSV report. By default, the semicolon (;) is used as a delimiter.

Related Commands

insert_observability_logic, report_data_gating_expression, set_powerpro_mode

Usage

The `report_data_gating` command reports data gating opportunities found on operators in the design. Additionally, it reports power changes, efficiency changes, and area changes for these operators.

Examples

Example 1: Generates the data gating report in text format and writes it to the `rdg_rep.rpt` file.

```
report_data_gating  -filename rdg_rep.rpt
```

Example 2: Generates a data gating report for all operators that have a data gating opportunity and writes the output to the `rdg_exp.csv` file. The colon(:) is specified as the delimiter.

```
report_data_gating  -filename rdg_exp.csv  -csv  -csv_delimiter :
```

report_data_gating_expression

Deprecated w.e.f. PowerPro 10.4. Instead of this command, use report_format -metric "Data Gating" -detail -filename <DGFileName.txt>.

report_data_gating_expression — Reports data gating domain information for operators with data gating opportunities in the design. Note that this command can be run only after the `insert_observability_logic` command if the global `opt_enable_data_gating` is enabled and the `generate_guidance` command has been run in the flow.

Synopsis

```
report_data_gating_expression  -filename <file_name>
```

-filename *<file_name>*

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

Related Commands

insert_observability_logic, report_data_gating, set_powerpro_mode

Usage

The **report_data_gating_expression** command reports the data gating domain information of the operators with data gating opportunity in the design.

Examples

Example 1: Generates the data gating expression report in text format and writes it to the `rdg_exp.rpt` file.

```
report_data_gating_expression  -filename rdg_exp.rpt
```

report_design

Deprecated w.e.f. PowerPro 10.4. Use the report_format command instead of this command. Alternatively, use the PowerPro Dashboard as the replacement of the information that this command provided.

report_design — Reports metrics for hierarchies in the design. Note that this command can only be called after `prototype_design`.

Synopsis

```
report_design -metric <metric_name1, metric_name2, ... >
    [ -start_instance <inst_name> ] [-filename <filename> ]
    [ -no_wrap ] [ -text | -xml | -csv ]
    [ -csv_delimiter <csv_delimiter> ]
```

-metric `<metric_name1, metric_name2, ... >`

Reports metric numbers for the specified `<metric_name>`. By default, metric numbers are reported for all hierarchies in the design unless the `-start_instance` option is specified. Legal values for the metric name include:

- enabled_flops (ef)
- strengthened_flops (sf)
- new_enabled_flops (nef)
- extra_area_added (eaa)
- register_power_savings (rps)
- register_power_saving_potential (rpsp)
- memory_power_savings (mps)
- memory_power_saving_potential (mpsp)
- total_memory_power (tmp)

- total_flop_power (tfp)

Note: Names in parentheses can be used instead of the full name. For example, to report metrics for enabled flops, specify `-metric ef` instead of `-metric enabled_flops`.

-start_instance *`<inst_name>`*

Specifies the name of the hierarchical instance for which the metric(s) numbers are to be reported. By default, numbers are reported starting with the top instance.

-filename *`<filename>`*

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

-no_wrap

By default, information is printed in fixed-width columns. If information for a field exceeds the column width, information continues on the next line under the same column. This option prevents information in columns from splitting across rows, making it easier to extract information from the report.

-text | -xml | -csv

Specifies the format of the report file to be generated. The information is reported in table format, with a choice of saving the report in text, XML and CSV format. By default, the report is generated as a plain text file.

Note: For CSV files, the semicolon(;) is used as the default delimiter. To specify another delimiter, use the `-csv_delimiter` option.

-csv_delimiter *`<csv_delimiter>`*

Specifies the delimiter for the CSV report. By default, the semicolon (;) is used as a delimiter.

Related Commands

insert_mem_clock_gating, insert_mem_observability_gating, insert_mem_sleep_gating, insert_mem_stability_gating, insert_observability_logic, insert_stability_logic

Usage

The `report_design` command can report the following metrics.

- `enabled_flops(ef)`: Reports percentage of enabled flops
- `strengthened_flops(sf)`: Reports number of strengthened flops
- `new_enabled_flops(nef)`: Reports number of new enabled flops
- `extra_area_added(eaa)`: Reports total extra area added
- `register_power_savings(rps)`: Reports total register power savings

- `register_power_saving_potential(rpsp)`: Reports total register power saving potential
- `memory_power_savings(mps)`: Reports memory power savings
- `memory_power_saving_potential(mpsp)`: Reports memory power saving potential
- `total_memory_power(tmp)`: Reports total memory power
- `total_flop_power(tfp)`: Reports total flop power

Note: While this command can be called in any of the modes, not all the metrics are displayed for all modes. Details are provided below.

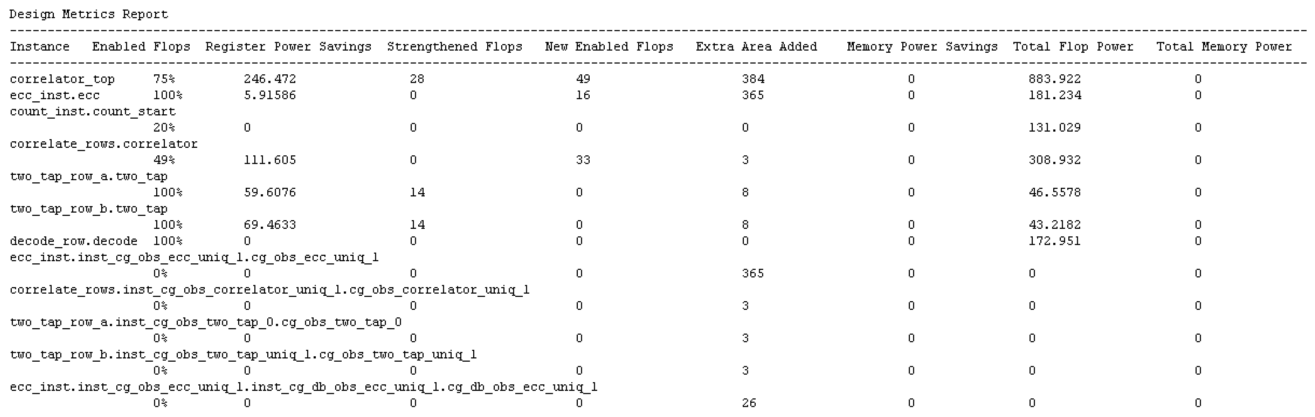
- In the PowerPro CG and MG modes, the metrics `ef`, `tmp` and `tfp` can only be reported after successful execution of the `prototype_design` command while the metrics `sf`, `nef`, `eaa`, `rps` and `mps` can only be reported after successful execution of sequential optimization commands.
- The metrics `rpsp` and `mpsp` are only available if the `generate_guidance` command has been run in the flow.
- The metrics `ef`, `rpsp`, `mpsp`, `tmp` and `tfp` can only be reported after successful execution of the `prototype_design` and `generate_guidance` commands while the metrics `sf`, `nef`, `eaa`, `rps` and `mps` can only be reported after successful execution of the sequential optimization commands.

Examples

Example 1: Reports metrics for all hierarchies in the design.

```
report_design -metric "ef rps sf nef eaa mps tfp tmp"
```

A sample report is shown below.

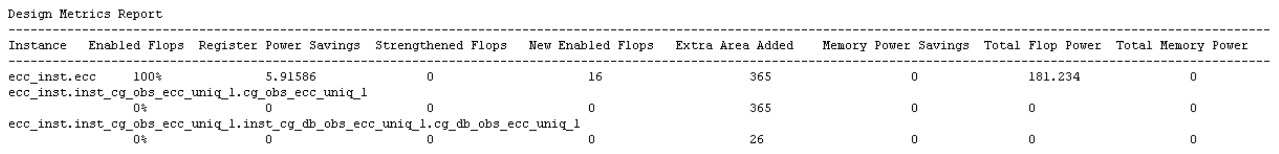


```
Design Metrics Report
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
Instance   Enabled Flops  Register Power Savings  Strengthened Flops   New Enabled Flops   Extra Area Added    Memory Power Savings  Total Flop Power   Total Memory Power
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
correlator_top     75%         246.472               28                    49                    384                   0               883.922              0
ecc_inst.ecc       100%        5.91586               0                     16                    365                   0               181.234              0
count_inst.count_start
                   20%         0                     0                     0                     0                     0               131.029              0
correlate_rows.correlator
                   49%         111.605               0                     33                    3                     0               308.932              0
two_tap_row_a.two_tap
                   100%        59.6076               14                    0                     8                     0               46.5578              0
two_tap_row_b.two_tap
                   100%        69.4633               14                    0                     8                     0               43.2182              0
decode_row.decode  100%        0                     0                     0                     0                     0               172.951              0
ecc_inst.inst_cg_obs_ecc_uniq_1.cg_obs_ecc_uniq_1
                   0%          0                     0                     0                     365                   0               0                    0
correlate_rows.inst_cg_obs_correlator_uniq_1.cg_obs_correlator_uniq_1
                   0%          0                     0                     0                     3                     0               0                    0
two_tap_row_a.inst_cg_obs_two_tap_0.cg_obs_two_tap_0
                   0%          0                     0                     0                     3                     0               0                    0
two_tap_row_b.inst_cg_obs_two_tap_uniq_1.cg_obs_two_tap_uniq_1
                   0%          0                     0                     0                     3                     0               0                    0
ecc_inst.inst_cg_obs_ecc_uniq_1.inst_cg_db_obs_ecc_uniq_1.cg_db_obs_ecc_uniq_1
                   0%          0                     0                     0                     26                    0               0                    0
```

Example 2: Reports metrics for the hierarchical instance `ecc_inst`.

```
report_design -metric "ef rps sf nef eaa mps tfp tmp" -start_instance
"ecc_inst"
```

A sample report is shown below.



```
Design Metrics Report
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
Instance    Enabled Flops  Register Power Savings  Strengthened Flops   New Enabled Flops   Extra Area Added    Memory Power Savings  Total Flop Power  Total Memory Power
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
ecc_inst.ecc     100%            5.91586                 0                    16                  365                   0                181.234              0
ecc_inst.inst_cg_obs_ecc_uniq_l.cg_obs_ecc_uniq_l
                 0%              0                       0                    0                   365                   0                0                    0
ecc_inst.inst_cg_obs_ecc_uniq_l.inst_cg_db_obs_ecc_uniq_l.cg_db_obs_ecc_uniq_l
                 0%              0                       0                    0                   26                    0                0                    0
```

report_design_toggle

report_design_toggle - Analyses the toggles in the design signals for the FSDB file to generate activity data. The command can be run only after the `read_fsdb` command and before the `report_power` command in the flow.

Synopsis

report_design_toggle { -interval_area <*`interval_value`*> | -num_slices <*`slice_count`*> }
[-start_instance <*`hier_inst_name`*>] [-depth <*`depth`*>]
[-toggle_waveform <*`file_name`*>] [-filename <*`filename`*>]

-interval_area <*interval_value*> | -num_slices <*slice_count*>

-interval_area divides the total simulation time into multiple intervals of size interval_value to calculate toggle count per interval. If the time unit is not specified, PowerPro automatically infers it from the FSDB file. Supported time units are: `s, ms, us, ns, ps, fs, as, zs,` and `ys`. The variable accepts only floating point and time values.

-num_slices divides the total simulation time into specified <slice_count> intervals. The size of the interval will be calculated according to the provided <slice_count>.

-start_instance <*hier_inst_name*>

Specifies the name of the hierarchical instance for which the toggle activity data must be reported. By default, it is reported for the top hierarchical instance.

-depth <*depth*>

Specifies the hierarchical depth till which the toggle activity data must be reported. By default, the toggle activity data is reported only for the top hierarchical instance. If the `-depth` option is not specified, toggle activity data is reported for the specified hierarchical instance. If this option is specified, the following rules apply:

1) <*depth*> takes the values {`0`, `1`, `2`, ...}

2) When <*depth*> is set to `0` and the `-start_inst` option is not specified, information is reported only for the top hierarchical instance. When <*depth*> is set to `0` and the `-start_inst` option is specified, information specific to <*hier_inst_name*> is reported.

-toggle_waveform <*filename*>

Generates a vcd file that contains the toggle activity data for all the hierarchical instances specified by the `-depth` or `-start_inst` options. By default, the report is generated for the top hierarchical instance. If the file with the same name exists, the contents of the file are overwritten. If the `-toggle_waveform` option is not specified, the file is saved at the following location `<work dir path>/report_toggle/report_toggle_for_<hier_inst_name>_<interval_value>.vcd`.

-filename <*filename*>

Specifies the name of the text file that will contain the toggling report for the specified hierarchal instance in below format. If the `-filename` option is specified, the value of the `-depth` option will be ignored. The file contains information in the following format:

Time Stamp Window	Toggle Count
...	...
...	...

If the file with the same name exists, the contents of the file are overwritten. If the `-filename` option is not specified, the file is saved at the following location `<work dir_path>/report_toggle/report_toggle_for_<hier_inst_name>_<interval_value>.txt`.

Related Commands

read_fsdb, report_flop_toggle

Usage

This command can be used to generate a text report and a histogram plot in PowerPro GUI for the specified design hierarchy. This histogram plot can be used to identify the high toggle activity region.

Examples

Example 1: Reports for the top hierarchical instance the toggle activity data that has an interval area of `1ns`. Note that a vcd and a text file are also generated at the default location `<work dir path>/report_toggle/`.

```
report_design_toggle -interval_area 1ns
```

Example 2: Reports for the specified instance `inst1.inst2` the toggle activity data that has an interval area of `100us`. Note that a vcd and a text file are also generated at the default location `<work dir path>/report_toggle/`.

```
report_design_toggle -interval_area 100us -start_inst inst1.inst2
```

Example 3: Reports for the top hierarchical instance and all the hierarchies at depth `1` the toggle activity data that have an interval area of `100us`. Note that a vcd and a text file are also generated at the default location `<work dir path>/report_toggle/`.

```
report_design_toggle -interval_area 100us -depth 1
```

Example 4: Reports for the specified instance `inst1.inst2` and all the hierarchies at depth `1` from that instance the toggle activity data that have an internal area of `100us`. Note that a vcd and a text file are also generated at the default location `<work dir path>/report_toggle/`.

```
report_design_toggle -interval_area 100us -start_inst inst1.inst2 -
depth 1
```

Example 5 : Reports for the top hierarchical instance the toggle activity data that has an interval area of `100us`. Note that, since the location is unspecified, the vcd file `design_activity` is generated in the working directory and a text file is generated at the default location `<work dir path>/report_toggle/`.

```
report_design_toggle -interval_area 100us -toggle_waveform
design_activity
```

Example 6 : Reports for the top hierarchical instance the toggle activity data that has an interval area of `100us`. Note that the vcd file is generated at the default location and the text file `top_toggle_info` is generated in the working directory, since the location is unspecified.

```
report_design_toggle -interval_area 100us -filename top_toggle_info
```

report_efficiency

report_efficiency — Reports the size and efficiency for the flops or hierarchical instance as computed by PowerPro. Note that this command can only be called after `prototype_design`.

Synopsis

```
report_efficiency [ -sequential ] [ -depth <depth> ]
    [ -ff_list <flop_list> |  -inst <hier_inst_name> ]
    [ -cond_enabled |  -always_enabled ] [-cg_min_size_based]
    [ -no_wrap]
    [ -text |  -xml |  -csv ] [ -csv_delimiter <csv_delimiter> ]
    [ -filename <filename> ]
```

-sequential

Specifies that the report include information specific to flops only.

-depth *<depth>*

Specifies the levels of hierarchy for which information is to be reported, with the following rules being applicable:

- <depth> takes the values {0, 1, 2,....}
- If `-inst` is specified, <depth> indicates the level of hierarchy upto which information will be reported.
- If `-inst` is specified along with -sequential, information for all the flops within the hierarchical instance specified upto <depth> will be reported.
- If `-depth` is not specified, information is reported for all hierarchical instances in the design.
- When *<depth>* is set to 0 and the -inst option is not specified, information is reported for the top module only. When <depth> is set to 0 and -inst is specified, information specific to <hier_inst_name> is reported.

Note: The `-depth` option cannot be specified with `-ff_list`. However, it can be specified with `-inst`.

-ff_list *<flop_list>* | -inst *<hier_inst_name>*

Specify `-ff_list` to report information for the flops listed by *<flop_list>*. When this option is specified, the `-sequential` option is ignored. *Note that the -depth option cannot be specified with* `-ff_list`.

Specify `-inst` to report information for the hierarchical instance specified by *<hier_inst_name>*.

If neither of these options are specified, the scope is set to the top module.

-cond_enabled | -always_enabled

Restricts the type of flops for which information is reported. By default, information is reported for all types of flops.

Specify `-cond_enabled`, to limit the scope of reporting to all flops that do not have an enable value of 1.

Specify `-always_enabled`, to limit the scope of reporting to all flops that have an enable value of 1.

Note: These options will be honored only when the `-sequential` or the `-ff_list` option has been specified.

-cg_min_size_based

Specify `-cg_min_size_based` to report the clock gating efficiency of the design hierarchies or sequential instance based on the global opt_cg_min_size.

-no_wrap

By default, information is printed in fixed -width columns. If information for a field exceeds the column width, information continues on the next line under the same column. This option prevents information in columns from splitting across rows, making it easier to extract information from the report.

-text | -xml | -csv

Specifies the format of the report file to be generated. The information is reported in table format, with a choice of saving the report in text, XML and CSV format. By default, the report is generated as a plain text file.

Note: For CSV files, the semicolon(;) is used as the default delimiter. To specify another delimiter, use the `-csv_delimiter` option.

-csv_delimiter *<csv_delimiter>*

Specifies the delimiter for the CSV report. By default, the semicolon (;) is used as a delimiter.

-filename *<filename>*

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

Related Commands

get_clock_gating_info, report_enable_logic, report_memory_efficiency

Usage

The **report_efficiency** command reports efficiency and size as computed by PowerPro for flops and instances in the design. By default, the report is generated for hierarchical instances.

Examples

Example 1: Reports efficiency numbers for all instances in the design.

```
report_efficiency
```

Example 2: Reports efficiency numbers for the hierarchical instance `I1.I2`, including instances within it.

```
report_efficiency  -inst I1.I2
```

Example 3: Reports efficiency numbers for all flops within the hierarchical instance `I1.I2`.

```
report_efficiency  -inst I1.I2  -sequential
```

Example 4: Reports efficiency numbers only for the top module. Efficiency numbers for child instances are not reported.

```
report_efficiency  -depth 0
```

Example 5: Reports efficiency numbers for all flops upto the 2nd level hierarchy of the top module.

```
report_efficiency  -sequential  -depth 2
```

Example 6: Reports efficiency numbers for the flops `f1, f2, f3` and `f4`.

```
report_efficiency  -ff_list {f1 f2 f3 f4}
```

report_enable_expression

Deprecated w.e.f. PowerPro 10.4. Instead of this command, use `report_format -metric "Clock Gating" -detail -filename <CGFileName.txt>`.

report_enable_expression — Reports CG Sequential Domain information corresponding to all the CG domains successfully committed during Sequential optimization; also reports Static Signal-Based Gating expressions identified in the design.

Synopsis

```
report_enable_expression [ -observability |  -stability |  -stability_c
    |  -stability_s |  -static |  -all ] [ -new_flops ]
    { -filename <report_file> }
```

-static | -observability | -stability | -stability_c | -stability_s | -all

Specifies the type of information written to the report.

Specify `-observability` to report the domains committed during observability based CG sequential optimization.

Specify `-stability` to report the domains committed during constant and symbolic stability based CG sequential optimization.

Specify `-stability_c` to report the domains committed during constant stability based CG sequential optimization.

Specify `-stability_s` to report the domains committed during symbolic stability based CG sequential optimization.

Specify `-static` to report gating expressions identified during Static Signal-Based Gating.

Specify `-all` to report all the domains committed during sequential optimization, as well as report Static Signal-Based Gating expressions identified in the design. This is the default.

-new_flops

Specifies that information about new CG -Hierarchy flops, contained in the enable expression, be printed at the end of the file *<report_file>*. By default, information is written to a new file by the name of *<report_file>*`.newflops.log`.

-filename*<report_file>*

Specifies the name of the report file. When this option is specified, a file by the name of *<report_file>*`.newflops.log` is also generated. This file contains details of the flops added by PowerPro during the sequential optimization phase.

Related Commands

report_static_signal_gating, report_enable_logic

Usage

The `report_enable_expression` command reports CG Sequential Domain Information corresponding to the domains that were committed during sequential optimization; also reports Static Signal-Based Gating expressions identified in the design.

Notes:

- The `-observability | -stability | -stability_c | -stability_s` options can only be called after successful execution of CG sequential optimization commands.
- The `-static` option can only be called after successful execution of the `insert_static_signal_gating` command.
- If the `report_enable_expression -all` command is called after `insert_static_signal_gating` but before any sequential optimization command is called, only Static Signal-Based Gating expressions will be reported.
- If the `report_enable_expression -all` command is called after `insert_static_signal_gating` and after sequential optimization commands are called then the report displays details for domains committed during sequential optimization as well as gating expressions generated during Static Signal-Based Gating.

Examples

Example 1: Generates a report containing CG Sequential Domain Information corresponding to all the domains that were committed during sequential optimization.

```
report_enable_expression  -filename report_file
```

Example 2: Generates a report containing CG Sequential Domain Information corresponding to all the domains committed during observability based sequential optimization.

```
report_enable_expression  -filename report_file  -observability
```

Example 3: Generates a report detailing gating expressions identified during Static Signal-Based Gating.

```
report_enable_expression  -filename report_file  -static
```

report_enable_logic

Deprecated w.e.f. PowerPro 10.4. Instead of this command, use `report_format -metric "Clock Gating"`.

report_enable_logic — Reports clock gating opportunities identified in the design by PowerPro for flops. Note that this command can only be issued after calling a CG sequential optimization command.

Synopsis

```
report_enable_logic [ -summary |  -enabled |  -disabled |  -comb ]
    [ -no_hier |  -ff_list <flop_list> ] [ -no_wrap ]
    [ -text |  -xml |  -csv |  -html <html> ]
    [ -csv_delimiter <csv_delimiter> ] [ -filename <filename> ]
    [ -sort <Efficiency Change|Power Change> ] [ -ascending ]
```

-summary | -enabled | -disabled | -comb

Specifies the type of information to be reported.

Specify `-summary` to generate information about newly generated enable logic in the design for each type of sequential optimization. The report also provides additional information about number of flops for each type, the clock -gating efficiency for those flops and the percentage of flops which were clock -gated.

Specify `-enabled` to generate information about flops that were clock -gated by PowerPro. This report includes the name of the enabled flop, information about the enable logic, the type of optimization performed as well as information about efficiency change and additional area.

Specify `-disabled` to generate information about flops that could not be clock -gated by PowerPro along with information like flop efficiency and size.

Specify `-comb` to report information about all user enabled flops, along with related information about enable logic, efficiency and area.

Note that these options can only be specified after CG sequential optimization commands with the exception of `-comb` which can be specified after `prototype_design`.

-no_hier | -ff_list *<flop_list>*

Specify `-no_hier` to report information for flops in the top hierarchy only.

Specify `-ff_list` to report information specific to the flops listed by *`<flop_list>`*.

-no_wrap

By default, information is printed in fixed -width columns. If information for a field exceeds the column width, information continues on the next line under the same column. This option prevents information in columns from splitting across rows, making it easier to extract information from the report.

-text | -xml | -csv | -html *`<html>`*

Specifies the format of the report file to be generated. The information is reported in table format, with a choice of saving the report in text, XML, CSV, and HTML format. By default, the report is generated as a plain text file.

Notes:

- The `-html` option requires the -enabled option to be specified.
- For CSV files, the semicolon(;) is used as the default delimiter. To specify another delimiter, use the `-csv_delimiter` option.

-csv_delimiter *`<csv_delimiter>`*

Specifies the delimiter for the CSV report. By default, the semicolon (;) is used as a delimiter.

-filename *`<filename>`*

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

-sort *`<sort_key>`*

Specifies that the generated report be sorted. Legal values for *`<sort_key>`* are `Efficiency Change` and `Power Change`. Specifying any other value will cause this option to be ignored i.e., the report will be generated but not sorted.

For example, this option must be specified as '-sort `'Power Change'`' instead of '-sort `Power Change`'.

-ascending

Specifies that the report be sorted in ascending order. By default, the report is sorted in descending order. This option requires the `-sort` option to be specified.

Related Commands

report_enable_expression

Usage

The **report_enable_logic** command reports clock gating opportunities identified by PowerPro.

Examples

Example 1: Reports information about all flops enabled by PowerPro during sequential optimization. Information is written to a text file by the name of `report1_g.txt`.

```
build_design input.v
prototype_design
insert_observability_logic  -effort high
report_enable_logic  -enabled -text  -filename report1_g.txt
```

In this example, to sort the generated output in descending order based on `Efficiency Change`, specify the following command:

```
report_enable_logic  -enabled  -text  -filename report1_g.txt
      -sort "Efficiency Change"
```

report_energy

report_energy — Reports internal, switching and leakage energy for a design. The command can be run only in the enhanced PA flow and can only be called after `report_power`.

Synopsis

```
report_energy [-instance_energy | -hierarchical | -
    detailed_hierarchical | -summary]
    [-start_instance <inst_name> ] [-depth <depth>]
    [-group [io_pad | memory | black_box | register | combinational |
    clock_network] ] [-clocks <clock_domain> ] [-no_wrap]
    [ -text | -csv ] [-csv_delimiter <csv_delimiter>]
    [-filename <filename> ] [-energy_unit <unit_type>]
    [-pa_html_report] [-output_dir <dir_name>]
```

-instance_energy | -hierarchical | -detailed_hierarchical | -summary

Specifies the scope of the report.

Specify `-instance_energy` to report energy information for instances only.

Specify `-summary` to generate a high-level report for the various energy groups. If `-start_instance` is also provided, information is reported for *`<inst_name>`* only.

Specify `-hierarchical` to display energy information for each block in hierarchical format. The `-depth` option can be used to limit the number of levels for which information is to be reported. With this option, internal, switching, leakage and total energy numbers are reported for each hierarchical block.

Specify `-detailed_hierarchical` to display energy detailed information for each block in hierarchical format. Hierarchies having area more than 10% of total design area will be reported by default. The `-depth` option can be used to limit the number of levels for which information is to be reported. The `-depth` option will be applied on full design and default behavior will be changed. With this option, internal, switching, leakage and total energy numbers are reported for each energy group of each hierarchical block.

-start_instance <*inst_name*>

Specifies the name of the hierarchical instance for which information is to be reported. By default, information is reported for all instances.

If specified with `-detailed_hierarchical` option, list of hierarchies can be passed.

Top hierarchy can be specified as `TOP_HIER` with other hierarchies in the list.

-depth <*depth*>

Specifies the depth of the instance to be reported. By default, information is reported for all levels. This option accepts any positive integer greater than `1`.

-group [io_pad | memory | black_box | register | combinational | clock_network]

Specifies that the report include information for a specific type of object only. The `-group` option can be used along with `-instance_energy`, `-hierarchical`, `-detailed_hierarchical` and `-summary` to further restrict the amount of information reported.

-clocks <*clock_domain*>

Specifies that the report includes information for the list of clock domains. The `-clocks` option can be used along with `-summary`, `-instance_energy`, `-start_instance` and `-hierarchical` to further restrict the amount of information reported.
Note: This option can be used only if PowerPro is launched using the `-pa` option or in Gate Level Power Analysis Flow. Only user clock domains can be specified with the `-clocks` option. The elements which are not under the user clock domain will be reported under clock-domain "virtual_clock".

-no_wrap

By default, information is printed in fixed-width columns. If information for a field exceeds the column width, information continues on the next line under the same column. This option prevents information in columns from splitting across rows, making it easier to extract information from the report.

-text | -csv

Specifies the format of the report file to be generated. The information is reported in table format, with a choice of saving the report in text and CSV format. By default, the report is generated as a plain text file.
Note: For CSV files, the semicolon(`;`) is used as the default delimiter. To specify another delimiter, use the `-csv_delimiter` option.

-csv_delimiter <*csv_delimiter*>

Specifies the delimiter for the CSV report. By default, the semicolon (`;`) is used as a delimiter.

-filename <*filename*>

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

-energy_unit <*unit_type*>

Specifies the base energy unit. Possible values are `fJ`, `pJ`, `nJ`, `uJ`, `mJ`, and `J`.

Default value is `uJ`.

[-pa_html_report]

Generates HTML page for each design hierarchy which contains energy summary report and energy information for all instances in the design hierarchy. All files are created in a directory specified using `-output_dir` option. Only options `–output_dir` and `-depth` are supported with this option. Name of the starting HTML page is `top.html`.

[-output_dir <*dir_name*>]

Specifies the directory in which HTML pages for design hierarchies are to be generated. This option can only be used with `-pa_html_report`. By default, all HTML pages are generated in the directory named `pa_html_report` in the work directory for the run.

Related Commands

report_power, set_powerpro_mode, prototype_design

Usage

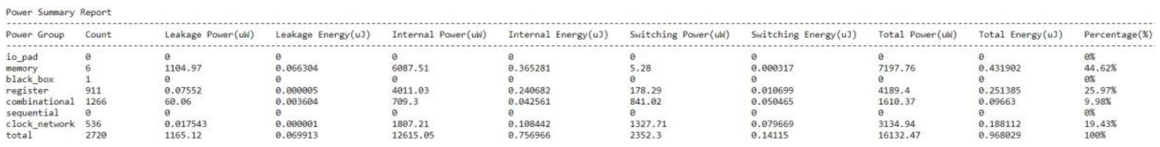
The **report_energy** command reports energy numbers for a design, based on switching activity. If switching activity has not been annotated using the `read_saif` command or, if not all nets are asserted, then in presence `-propagate on` option it will estimate the switching activity of unasserted nets in the design based on the switching activity of the nets in their transitive fanin cones. If primary inputs have no switching activity asserted then they default to a toggle rate of `0.005/ns` and a probability of `0.5`.

Examples

Example 1: Reports summary of energy for the design, along with the environment setup information.

```
powerpro> report_energy -summary
```

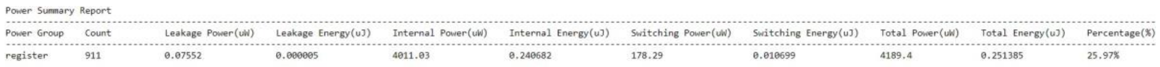
A sample report is shown below.



```
Power Summary Report
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
Power Group    Count          Leakage Power(uW)    Leakage Energy(uJ)    Internal Power(uW)    Internal Energy(uJ)    Switching Power(uW)    Switching Energy(uJ)    Total Power(uW)    Total Energy(uJ)    Percentage(%)
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
io_pad         0              0                    0                     0                     0                      0                      0                       0                  0                   0%
memory         6              1104.97              0.066304              6087.51               0.365281               5.28                   0.000317                7197.76            0.431902            44.62%
black_box      1              0                    0                     0                     0                      0                      0                       0                  0                   0%
register       911            0.07552              0.000005              4011.03               0.240682               178.29                 0.010699                4189.4             0.251385            25.97%
combinational  1266           60.06                0.003604              709.3                 0.042561               841.02                 0.050465                1610.37            0.09663             9.98%
sequential     0              0                    0                     0                     0                      0                      0                       0                  0                   0%
clock_network  536            0.017543             0.000001              1807.21               0.108442               1327.71                0.079669                3134.94            0.188112            19.43%
total          2720           1165.12              0.069913              12615.05              0.756966               2352.3                 0.14115                 16132.47           0.968029            100%
```

Example 2: Prints a summary for the group 'register'.

```
powerpro> report_energy -summary -group register
```



```
Power Summary Report
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
Power Group    Count          Leakage Power(uW)    Leakage Energy(uJ)    Internal Power(uW)    Internal Energy(uJ)    Switching Power(uW)    Switching Energy(uJ)    Total Power(uW)    Total Energy(uJ)    Percentage(%)
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
register       911            0.07552              0.000005              4011.03               0.240682               178.29                 0.010699                4189.4             0.251385            25.97%
```

report_flop_toggle

report_flop_toggle — Generates for the specified design hierarchy a report and a histogram plot that illustrate the correlation between the bit-level flops and toggle density ranges. If the hierarchy is not specified, it will generate a report for the top-level design.

Synopsis

```
report_flop_toggle [-current_design <instance_name>] [-min <val>] [-max
    <val>] [-resolution <val>] [-filename <filename>] [ -text | -csv ]
    [-plot] [-exclude_list <list_of_objects>] [-exclude_list_file
    <file_name>]
```

-current_design <instance_name>

Specifies the design hierarchy for which the report has to be generated.

-min <val>

Specifies the value from which the x-axis of the histogram plot will start. The x-axis depicts the toggle density range. Default value is 0, that is, the x-axis of the histogram plot will start from the value 0.

-max <val>

Specifies the value at which the x-axis of the histogram plot will end. Among the toggle density values of all the flops, default value is the ceiling of the maximum toggle density value.

-resolution <val>

Specifies the step-size or resolution of the x-axis or the td range. Default value is 0.01.

-filename <filename>

Specifies the name of the report file that must be created. If a file with the same name exists, the contents of the file are overwritten. If this option is not specified, a report is generated in the file named histogram.log.

-text | -csv

Specifies the format in which the report file will be generated. The information is reported in table format, with a choice of saving the report in text and CSV format. By default, the report is generated in text format.

Note: For CSV files, the semicolon(;) is used as the default delimiter.

-plot

Generates the histogram plot, and a gnuplot script gnuplot.gp is saved in the current directory. The saved gnuplot script can be used to generate the histogram plot later.

-exclude_list <list_of_objects>

Specifies a list of the hierarchical names of the flops for which the report or the histogram plot must not be generated.

-exclude_list_file <file_name>

Specifies the name of the file which contains the information about the flops for which the report or the histogram plot must not be generated.

Related Commands

get_sa, read_fsdb, read_saif

Usage

Generates for the specified design hierarchy a report and a histogram plot that illustrate the correlation between the bit-level flops and toggle density ranges. If the hierarchy is not specified, it will generate a report for the top-level design.

Examples

Example 1: Reports the toggle density ranges and generates a histogram plot for all the flops at the top-level.

```
powerpro> report_flop_toggle -plot
```

Example 2: Reports toggle density ranges for the design hierarchy instance ecc_inst, for which the x-axis of the histogram plot will start from 0.06 and end at 0.12, and the resolution of the x-axis is 0.001.

```
powerpro> report_flop_toggle -current_design ecc_inst -min 0.06 -max
0.12 -resolution 0.001 -plot
```

Example 3: Reports toggle density ranges for the design hierarchy instance ecc_inst, in which the report for the flops ecc_inst.mode_t and ecc_inst.ecc must not be generated.

```
powerpro> report_flop_toggle -current_design ecc_inst -exclude_list
{ecc_inst.mode_t ecc_inst.ecc}
```

report_format

report_format — Displays and generates detailed reports for all the PowerPro metrics in the specified format. The command can also generate customized reports using filters.

Synopsis

```
report_format  [-group <group_name> | -metric <metric_name> ]
    [ -text | -csv | -csv_delimiter <delimiter> ]
    [ -html ]  [ -no_wrap ] [ -fold ] [ -no_escape_name ]
    [ -precision <precision_value> ]
    [ -detail ]  [ -height <max_height> ] [ -depth <max_depth> ]
    [ -count <number_of_rows> ] [ -configuration <configuration_name> ]
    [ -dir <golden_work_dir> ]
    [ -filename <name> ]  [ -output_dir <name> ]
```

-group <*group_name*> | -metric <*metric_name*>

Use the `-group` option to restrict the generation of detailed reports only for the metrics that belong to the specified group.

Use the `-metric` option to restrict the generation of detailed reports only for the specified metric.

-text | -csv | -csv_delimiter <*delimiter*>

Use the `-text` option to generate the reports in the text format.

Use the `-csv` option to generate the reports in the csv format.

Use the additional option, `-csv_delimiter` with the `-csv` option to specify a custom delimiter to be used in the report csv file.

Note: The options `-text` and `-csv` are mutually exclusive.

-html

Generates a script that allows offline viewing of reports in the HTML format.

To view the reports in the HTML format, run `report_format` with `-html`. This option generates a script, `start_dynamic_reports_server.sh`, to start an on-demand Dynamic HTML Reports server.

For viewing the PowerPro reports offline, run the script and open the generated URL in a web browser that supports HTML5, such as Firefox 40+, Chrome 37+, and IE 11+. Then, navigate to the details you want to view.

Note: Do not use the `-html` option with any other options except the `-output_dir` option.

-no_wrap

By default, information is printed in fixed-width columns. If the information in a field exceeds the column width, that information continues on the next line in the same column. Use this option to prevent information in columns from splitting across rows, which makes information extraction from the report easier.

-fold

By default, all rows, including child rows, are printed for a report. Use this option to prevent child rows from being printed.

-no_escape_name

By default, all the names are printed with escape characters. Use this option if the names must be printed without escaping.

-precision <*precision_value*>

Specifies the precision for all the floating numbers in the reports.

-detail

Generates a separate detail file for each report along with its domain information. A detail file is available as *<report>_Details.txt*.

-height <*max_height*>

Specifies the maximum distance from the leaf hierarchy in bottom-up manner till which the hierarchical data will be reported.

Note : The `-height` option and the `-depth` option are mutually exclusive.

-depth <*max_depth*>

Specifies the maximum distance of hierarchy from the root node in top-down manner till which the hierarchical data will be reported.

Note : The `-depth` option and the `-height` option are mutually exclusive.

-count <*number_of_rows*>

Specifies the number of rows to be printed in the reports.

-configuration <*configuration_name*>

> Specifies the name of the configuration for which reports are to be generated. Few available configurations are DEFAULT | PADEBUG | PADIFF. If this option is not specified, DEFAULT configuration will be picked to generate reports. PADEBUG and PADIFF are enhanced PA Debugging configurations.

-dir <*golden_work_dir*>

> Specifies the path of the golden work directory against which PADIFF configuration reports will be created. This option can only be used with PADIFF configuration.

-filename <*filename*>

> Specifies the name of the report file to create. If the file exists, its contents are overwritten. If this option is not specified, the report is sent to the standard output.
>
> Note: The `-filename` option can be used with the `-metric` or `-list` option.

-output_dir *`<name>`*

> Specifies the name of the directory in which the output files for the selected reports must be stored. By default, the name of the output directory is `report_format_`*`<count>`*, where the value of *`count`* starts with `0` and increments with every command run.
>
> Note: The `-output_dir` option cannot be used with the `-metric` option.

Related Commands

show_analyzer

Usage

Displays the specified reports in the given format.

Examples

Example 1: Generate reports in the text format and store them in the `all_text_reports` directory.

```
 report_format -text -output_dir all_text_reports
```

Example 2: Generate the `Early Design Checks` reports in the csv format using the csv_delimiter `delimiter` and store them in the `early_design_checks_csv_reports_with_delimiter` directory.

```
report_format -group "Early Design Checks" -csv -csv_delimiter
delimiter -output_dir early_design_checks_csv_reports_with_delimiter
```

Example 3: Generate the `User Enabled Flops` report in the text format and store it in the `usr_enabled_flop.txt` file.

```
report_format -metric "User Enabled Flops" -text -filename
usr_enabled_flop.txt
```

report_globals

report_globals — Reports current global variable settings.

Synopsis

`report_globals [-no_wrap] [-filename <filename> ]`

-no_wrap

By default, information is printed in fixed-width columns. If information for a field exceeds the column width, information continues on the next line under the same column. This option prevents information in columns from splitting across rows, making it easier to extract information from the report.

-filename *`<filename>`*

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, current global settings are written to the standard output.

Related Commands

set_global

Usage

This command reports the current global variable settings.

Examples

An extract from the report is shown below.

```
Global Variable Settings

-----------------------------------------------------------------------------
Name                                           Value        Category
-----------------------------------------------------------------------------
alert_color                                    light_red    Log File Formatting
maximum_iter_count                             1024         Design Read & Write
maximum_recurse_count                          100          Design Read & Write
message_wrap_at                                79           Log File Formatting
opt_add_async_powerpro_reset                   0            Constraints & Setup
opt_assume_default_area_for_memories_with_zero_area
                                               0            Miscellaneous
opt_cg_min_size                                4            Constraints & Setup
```

Example 2: Writes out current global settings to the file `globals.txt` without wrapping columns to the next line.

```
report_globals  -no_wrap  -filename globals.txt
```

An extract from the file `globals.txt` is shown below.

```
Global Variable Settings
------------------------------------------------------------------------------
Name                                           Value        Category
------------------------------------------------------------------------------
alert_color                                    light_red    Log File Formatting
maximum_iter_count                             1024         Design Read & Write
maximum_recurse_count                          100          Design Read & Write
message_wrap_at                                79           Log File Formatting
opt_add_async_powerpro_reset                   0            Constraints & Setup
opt_assume_default_area_for_memories_with_zero_area  0              Miscellaneous
opt_cg_min_size                                4            Constraints & Setup
```

report_hierarchy

report_hierarchy — Reports design hierarchy information. Note that this command can only be called after `build_design`.

Synopsis

```
report_hierarchy [ -module <module> ] [ -depth <depth> ]
     [ -start_inst <start_inst> ] [ -inst |  -noinst ]
     [ -skip_black_box_instances] [ -list_format]
     [ -filename <filename> ]
```

-module *<module>*

Specifies the name of the module from where reporting begins. By default, information is reported for all top -level modules. Note that the `-start_inst` option cannot be specified with this option.

-depth *<depth>*

Specifies the depth upto which hierarchy information is to be reported. Reports the hierarchy up to the specified depth. A depth of 0 displays information only for the top level hierarchy. By default, hierarchy information is reported for all levels.

-start_inst *<start_inst>*

Specifies the name of the instance from which reporting should begin. Note that this option cannot be specified with the `-module` option.

-inst | -noinst

Specifies whether or not the names of the instances are to be displayed along with the module name.

Specify `-inst` to display instance names next to the modules name. This is also the default.

Specify `-no_inst` to disable display of instance names next to the module names.

-skip_black_box_instances

Skips reporting for blackboxed instances.

-list_format

Specifies that the information be returned as a Tcl list. By default, information is printed in the form of a report.

-filename *<filename>*

Specifies the name of the output file. If not specified, the report is sent to the standard output.

Related Commands

find, report_memory_instances

Usage

The **report_hierarchy** command reports design hierarchy information.

A few sample reports are shown below. Results of Examples 2 and 3 are based on the hierarchy shown in Example 1.

Example 1: Reports design hierarchy information in tree format.

```
powerpro> report_hierarchy

__Top_Node__(C)
    b2_c(B)
        a2_b(A)
        a1_b(A)
    a_c(A)
    b1_c(B)
        a2_b(A)
        a1_b(A)
```

Example 2: Reports design hierarchy information starting with the instance `b2_c`.

```
powerpro> report_hierarchy  -start_inst b2_c

b2_c(B)
   a2_b(A)
   a1_b(A)
```

Example 3: Reports design hierarchy information for 1 level of hierarchy (`depth = 1`).

```
powerpro> report_hierarchy   -depth 1

__Top_Node__(C)
    b2_c(B)
    a_c(A)
    b1_c(B)
```

report_high_toggling_signals

Deprecated w.e.f. PowerPro 10.4. Instead of this command, use `report_format -metric "High Toggling Signals"`.

report_high_toggling_signals — Reports percentage toggle activity found on the signals connected to hierarchies in the design. Note that this command can only be called after `prototype_design` if the `generate_guidance` command has been run in the flow.

Synopsis

```
report_high_toggling_signals [ -filename <filename> ] [ -no_wrap ]
    [ -text |  -xml |  -csv ] [ -csv_delimiter <csv_delimiter> ]
```

-filename *<filename>*

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

-no_wrap

By default, information is printed in fixed -width columns. If information for a field exceeds the column width, information continues on the next line under the same column. This option prevents information in columns from splitting across rows, making it easier to extract information from the report.

-text | -xml | -csv

Specifies the format of the report file to be generated. The information is reported in table format, with a choice of saving the report in text, XML and CSV format. By default, the report is generated as a plain text file.

Note: For CSV files, the semicolon(;) is used as the default delimiter. To specify another delimiter, use the `-csv_delimiter` option.

-csv_delimiter *<csv_delimiter>*

Specifies the delimiter for the CSV report. By default, the semicolon (;) is used as a delimiter.

Related Commands

report_mem_inputs_redundant_toggle

Usage

The `report_high_toggling_signals` command reports percentage toggle activity on the signals connected to hierarchies in the design. Additionally, it reports width of the signal, driver of the signal, type of driver and power for these signals.

Note: The numbers reported by the `report_high_toggling_signals` command do not get updated post sequential optimization. It is recommended that this command be called before any sequential optimization command. This limitation will be fixed in a future release.

Examples

Example 1: Generates a high toggling signals report in text format and writes it to the `rhts_rep.rpt` file.

```
report_high_toggling_signals  -filename rhts_rep.rpt
```

Example 2: Generates a high toggling signals report and writes the output to the `rhts_exp.csv` file. The colon(:) is specified as the delimiter.

```
report_high_toggling_signals  -filename rhts_exp.csv  -csv
    -csv_delimiter :
```

report_html_format

Deprecated w.e.f. PowerPro 10.2. To generate HTML reports, use the command `report_format -html` instead.

report_html_format — Displays the available reports in HTML5 format, which can be viewed in any HTML5 and Javascript-enabled web browser. The output html page is known as `index.html`. This command can be run only after the `prototype_design` and the `report_power` commands have been run.

By default, index.html displays all the available reports that have been generated in the flow. This includes Design Statistics, Design Power, and so on.

Synopsis

```
report_html_format [ -combinational ] [ -sequential ] [ -microarch ] [
-output_dir <name> ]
```

-combinational | -sequential | -microarch

Specifies the filter to be applied among all the available redundancy reports.

If you specify the `-combinational` argument, only the combinational redundancy reports are displayed in HTML5 format. These include redundant mux activity, high-toggling buses, memory interface toggle, and so on. This option can be used only if the `generate_guidance` command has been run in the PowerPro flow.

If you specify the `-sequential` argument, only the sequential redundancy reports are displayed in HTML5 format. These include clock gating, data gating, redundant reset optimization, and so on. This option can be run only if observability-based CG-optimization has been done in the flow.

If you specify the `-microarch` argument, only the micro-architectural redundancy reports are displayed in HTML5 format. These include block-level clock gating, shift register to circular buffer, and so on. This option can be used only if the `generate_guidance` command has been run in the PowerPro flow and PowerPro Exploration license is available.

-output_dir *<name>*

Specifies the name of the directory in which the HTML files for the selected reports must be stored. By default, the name of the output directory is `html_format_<cnt>`, where the value of `cnt` starts with `0` and increments with every command run.

Related Commands

report_enable_logic, report_data_gating, report_high_toggling_signals, report_mem_inputs_redundant_toggle, report_redundant_reset_flops

Usage

The `report_html_format` command displays in HTML format the combinational, sequential, and micro-architectural redundancies in the design. The first html page is known as `index.html`.

Note: The recommended web browser for viewing HTML reports is `Chrome 52` or above.

The HTML reports can also be viewed on the following web browsers, with the specified browser versions or above: `Chrome 23`, `Firefox 18`, `Opera 12.10`, `Internet Explorer 11`, `Safari 8.0`, and `Edge 12`.

Examples

Example 1: Generates HTML files in the `comb_html_report` directory for the combinational redundancies available in the design.

```
report_html_format -combinational -output_dir comb_html_report
```

Example 2: Generates HTML files in the `seq_html_report` directory for the sequential redundancies available in the design.

```
report_html_format -sequential -output_dir seq_html_report
```

Example 3: Generates HTML files in the `html_report` directory for both combinational and sequential redundancies available in the design.

```
report_html_format -combinational -sequential -output_dir html_report
```

Example 4: Generates HTML files in the `html_report` directory for all redundancies available in the design.

```
report_html_format -output_dir html_report
```

Example 5: Generates HTML files in the `micro_html_report` directory for the micro-architectural redundancies available in the design.

```
report_html_format -microarch -output_dir micro_html_report
```

report_ip_metrics

Deprecated w.e.f. PowerPro 10.4. Instead of this command, use `report_format -ip`.

report_ip_metrics — The `report_ip_metrics` command reports the modules that are potentially consuming high power in the given design context.

Note that this command can be run only after `prototype_design` and `report_power` commands have been run.

Synopsis

```
report_ip_metrics  [-output_dir <output_dir>]  [-precision  <precision>]
[-no_wrap] [ -text | -xml | -csv ] [-csv_delimiter <csv_delimiter>] [-
filename <filename>]
```

-output_dir <*output_dir*>

Specifies the path of the directory where the report will be generated.

-precision <*precision*>

Specifies the number of decimal places till which the values in the report must be printed. For example, the number `123.45` has a precision of `2`.

-no_wrap

By default, the information is printed in fixed-width columns. If the information for a field exceeds the column width, it continues on the next line under the same column. This option prevents the information in columns from splitting across rows, making it easier to extract it from the report.

-text | -xml | -csv

Specifies the format in which the report must be generated. The information is printed in table format, with a choice of saving the report in text, XML and CSV formats. By default, the report is generated as a plain text file.

Note: For CSV files, the semicolon (;) is used as the default delimiter. To specify a different delimiter, use the `-csv_delimiter` option.

-csv_delimiter <*csv_delimiter*>

Specifies the delimiter for the CSV report. Note that semicolon (;) is used as the default delimiter.

-filename <*filename*>

Specifies the name of the file in which the report is printed. If the file with the same name exists, the contents of the file are overwritten. If this option is not specified, the report is displayed on the standard output.

Related Commands

report_html_format, report_hierarchy

Usage

In this report, power saving potential is reported based on the modules in the design. Therefore, it is easy to identify the module that is wasting the maximum power.

Note that the design objects that are wasting maximum power can be identified by using the different PowerPro guidance reports. However, in some cases, the reported design object is not wasting the maximum power, but, is instantiated multiple times in the design, which collectively consumes more power. In such cases, a single change in the

RTL can help save power across all instances of the object. To understand the RTL changes that should be performed first, this report can be used in GUI.

Examples

Example 1: Displays the report on PowerPro command line interface.

```
report_ip_metrics
```

Example 2: Displays the report in text format in the `ip_data.txt` file, where the values can be printed up to two decimal places.

```
report_ip_metrics -text -precision 2 -filename ip_data.txt
```

report_mem_enable_expression

Deprecated w.e.f. PowerPro 10.4. Instead of this command, use `report_format -metric "Memory Gating" -detail -filename <MGFileName.txt>`.

report_mem_enable_expression — Reports MG Sequential Domain information corresponding to all the MG domains that were successfully committed during memory sequential optimization. Note that this command can only be called after calling an MG sequential optimization command.

Synopsis

```
report_mem_enable_expression
    [ -observability |  -stability |  -sleep |  -all ] [ -new_flops]
    { -filename <report_file> }
```

-observability | -stability | -sleep | -all

Specifies the type of MG sequential domain information written to the report.

Specify `-observability` to report information for domains committed during observability based MG sequential optimization.

Specify `-stability` to report information for domains committed during constant and symbolic stability based MG sequential optimization.

Specify `-sleep` to report information for domains committed during light sleep based MG sequential optimization.

Specify `-all` to report information for all domains committed during MG sequential optimization. This is the default.

-new_flops

Specifies that information about new MG -Hierarchy flops, contained in the enable expression, be printed at the end of the file *`<report_file>`*. By default, information is written to a new file by the name of *<report_file>*`.newflops.log`.

-filename *<report_file>*

Specifies the name of the report file. When this option is specified, a file by the name of *<report_file>*`.newflops.log` is also generated. This file contains details of the flops added by PowerPro during `insert_mem_stability_gating`.

Related Commands

report_memory_gating

Usage

The **report_mem_enable_expression** command reports MG Sequential Domain Information corresponding to the domains that were committed during sequential optimization.

Examples

Example 1: Reports information corresponding to all the domains committed during MG Sequential optimization.

```
report_mem_enable_expression  -filename report_file
```

Example 2: Reports information corresponding to all the domains committed during observability based MG Sequential optimization.

```
report_mem_enable_expression  -filename report_file  -observability
```

report_mem_inputs_redundant_toggle

Deprecated w.e.f. PowerPro 10.4. Instead of this command, use `report_format -metric "Redundant Memory Address Toggles"` and `report_format -metric "Redundant Memory Data Toggles"`.

report_mem_inputs_redundant_toggle — Reports percentage of redundant and wasted toggle activity found on the clock, address, and data port of memories in the design. Note that this command can only be called after `prototype_design` if the `generate_guidance` command has been run in the flow.

Synopsis

```
report_mem_inputs_redundant_toggle
    [ -filename <filename>] [ -no_wrap] [ -text | -csv ]
    [ -csv_delimiter <csv_delimiter>]
```

-filename *<filename>*

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

-no_wrap

By default, information is printed in fixed -width columns. If information for a field exceeds the column width, information continues on the next line under the same column. This option prevents information in columns from splitting across rows, making it easier to extract information from the report.

-text | -csv

Specifies the format of the report file to be generated. The information is reported in table format, with a choice of saving the report in text and CSV format. By default, the report is generated as a plain text file.

Note: For CSV files, the semicolon(;) is used as the default delimiter. To specify another delimiter, use the `-csv_delimiter` option.

-csv_delimiter *<csv_delimiter>*

Specifies the delimiter for the CSV report. By default, the semicolon (;) is used as a delimiter.

Related Commands

report_high_toggling_signals

Usage

The `report_mem_inputs_redundant_toggle` command reports percentage of redundant and wasted toggle activity found on the clock, address and data port of memories in the design.

Note: The numbers reported by the `report_mem_inputs_redundant_toggle` command do not get updated post sequential optimization. It is recommended that this command be called before any sequential optimization command. This limitation will be fixed in a future release.

Examples

Example 1: Generates the redundant memory input toggle activity report in text format and writes it to the `rrmit_rep.rpt` file.

```
report_mem_inputs_redundant_toggle  -filename rrmit_rep.rpt
```

Example 2: Generates a redundant memory input toggle activity report and writes the output to the `rrmit_exp.csv` file. The colon(:) is specified as the delimiter.

```
report_mem_inputs_redundant_toggle  -filename rrmit_exp.csv  -csv
     -csv_delimiter ,
```

report_memory_efficiency

report_memory_efficiency — Reports efficiency numbers for Memory Enables and LS ports in the design, as computed by PowerPro. Note that this command can only be called after `prototype_design`.

Synopsis

```
report_memory_efficiency [ -depth <depth> ]
    [ -mem_list <memory_list> | -inst <hier_inst_name> ]
    [ -observability | -stability | -sleep | -all ]
    [ -enable_ports | -ls_ports | -all_ports ] [ -no_wrap ]
    [ -text |  -xml | -csv ] [ -csv_delimiter <csv_delimiter> ]
    [ -filename <filename> ]
```

-mem_list *<memory_list>* | -inst *<hier_inst_name>*

Specifies the scope of the report.

Specify `-mem_list` to report information for the memories specified by *<memory_list>*. All other options, defining the scope of memories to be reported, are ignored when this option is specified. *Note that the* `-depth` *option cannot be specified with* `-mem_list`.

Specify `-inst` to report information specific to Memory Enables and LS ports of memories within the hierarchical instance *<hier_inst_name>*. If not specified, the scope is set to the top module.

-depth *<depth>*

Specifies the levels of hierarchy for which information is to be reported, with the following rules being applicable:

- *<depth>* takes the values {0, 1, 2,....}
- If `-inst` is specified, *<depth>* indicates the level of hierarchy upto which information will be reported.
- If `-depth` is not specified, information is reported for all hierarchical instances in the design.
- When *<depth>* is set to `0` and the `-inst` option is not specified, information is reported for the top module only. When *<depth>* is set to `0` and `-inst` is specified, information specific to *<hier_inst_name>* is reported.

Note: This option cannot be specified with `-mem_list`. However, it can be specified with `-inst`.

-observability | -stability | -sleep | -all

Restricts the scope of the report.

Specify `-observability` to report only those memories for which observability based moves were identified during MG sequential optimization.

Specify `-stability` to report only those memories for which stability based moves were identified during MG sequential optimization.

Specify `-sleep` to report only those memories for which light sleep based moves were identified during MG sequential optimization.

Specify `-all` to report information for all memories in the design. This is the default.

-enable_ports | -ls_ports | -all_ports

Specifies the type of ports for which information is to be reported.

Specify `-enable_ports` to report information corresponding to Memory Enable ports only.

Specify `-ls_ports` to report information corresponding to Memory LS ports only.

Specify `-all_ports` to report information for both Memory Enable and LS ports in the design. This is the default.

-no_wrap

By default, information is printed in fixed -width columns. If information for a field exceeds the column width, information continues on the next line under the same column. This option prevents information in columns from splitting across rows, making it easier to extract information from the report.

-text | -xml | -csv

Specifies the format of the report file to be generated. The information is reported in table format, with a choice of saving the report in text, XML and CSV format. By default, the report is generated as a plain text file.

Note: For CSV files, the semicolon(;) is used as the default delimiter. To specify another delimiter, use the `-csv_delimiter` option.

-csv_delimiter *<csv_delimiter>*

Specifies the delimiter for the CSV report. By default, the semicolon (;) is used as a delimiter.

-filename *<filename>*

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

Related Commands

report_efficiency, report_memory_gating, report_memory_instances, report_memory_power

Usage

The **report_memory_efficiency** command reports efficiency numbers for Memory Enable and LS ports in the design as computed by PowerPro.

Examples

Example 1: Reports efficiency numbers for Memory Enables and LS ports of all memories in the design.

```
report_memory_efficiency
```

Example 2: Reports efficiency numbers for Memory Enables and LS ports of all memories within the hierarchical instance I1.I2.

```
report_memory_efficiency  -inst I1.I2
```

Example 3: Reports efficiency numbers for Memory Enables and LS ports of all memories in the top module only. Efficiency numbers for memories within the child instances are not reported.

```
report_memory_efficiency  -depth 0
```

Example 4: Reports efficiency numbers for Memory Enables and LS ports of the memories `I1.mem1`, `I1.I2.mem`, `mem1`, and `mem2`.

```
report_memory_efficiency  -mem_list {I1.mem1 I1.I2.mem mem1 mem2}
```

report_memory_gating

Deprecated w.e.f. PowerPro 10.4. Instead of this command, use `report_format -metric "Memory Gating"`.

report_memory_gating — Reports memory gating opportunities identified by PowerPro in the design.

Synopsis

```
report_memory_gating [ -mem_list <memory_list>]
    [ -summary | -all | -observability | -stability | -sleep ]
    [ -text | -xml | -csv | -html <html> ] [ -no_wrap]
    [ -csv_delimiter <csv_delimiter>] [ -filename <filename>]
```

-mem_list *<memory_list>*

Specifies the list of memories for which the report is to be generated. By default, the report is generated for all PowerPro-gated memories in the design.

-summary | -all | -observability | -stability | -sleep

Specifies the type of information to include in the report. If this option is not specified, PowerPro generates a summary report.

Specify `-summary` to report a summary of newly generated memory enable logic.

Specify `-all` to generate a detailed report for all memories in the design.

Specify `-observability` to generate a detailed report for those memories on which observability based memory optimization was performed.

Specify `-stability` to generate a report for those memories on which stability based memory optimization was performed.

Specify `-sleep` to generate a report for those memories on which light sleep based memory optimization was performed.

-no_wrap

By default, information is printed in fixed-width columns. If information for a field exceeds the column width, information continues on the next line under the same column. This option prevents information in columns from splitting across rows, making it easier to extract information from the report.

-text | -xml | -csv | -html *<html>*

Specifies the format of the report file to be generated. The information is reported in table format, with a choice of saving the report in text, XML, CSV and HTML format. By default, the report is generated as a plain text file.

Note: For CSV files, the semicolon(;) is used as the default delimiter. To specify another delimiter, use the `-csv_delimiter` option.

-csv_delimiter *<csv_delimiter>*

Specifies the delimiter for the CSV report. By default, the semicolon (;) is used as a delimiter.

-filename *<filename>*

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

Related Commands

insert_mem_observability_gating, insert_mem_stability_gating, insert_mem_sleep_gating, report_memory_efficiency, report_memory_instances, report_memory_power

Usage

The `report_memory_gating` command reports information about memory gating opportunities identified by PowerPro.

Examples

Example 1: Generates a report for all PowerPro-gated memories in the design.

```
build_design test.v
...
...
prototype_design
insert_mem_observability_gating
report_memory_gating  -all -text -filename report1.rpt
```

report_memory_instances

Deprecated w.e.f. PowerPro 10.4. Use the command `report_format -metric "Memory Info"` instead.

report_memory_instances — Reports properties for the memories in the design. Note that this command can only be called after `prototype_design`.

Synopsis

```
report_memory_instances
    [ -mem_list <memory_list> | -inst <hier_inst_name> ]
    [ -depth <depth> ] [ -no_wrap ] [ -text |  -xml |  -csv ]
    [ -csv_delimiter <csv_delimiter> ] [ -filename <filename> ]
```

-mem_list *`<memory_list>`* | -inst *`<hier_inst_name>`*

Specifies the scope of the report.

Specify `-mem_list` to report information only for the memories specified by `memory_list`. All other options defining the scope of the report are ignored when this option is specified. Note that the `-depth` option cannot be specified with `-mem_list`.

Specify `-inst` to report information for the memories within the hierarchical instance *`<hier_inst_name>`*. If not specified, the scope is set to the top module.

-depth *`<depth>`*

Specifies the depth of the instance to be reported. By default, properties are reported for all levels. When this option is specified, the following rules apply:

- `<depth>` takes the values `{0, 1, 2,....}`
- If `-inst` is specified, *`<depth>`* indicates the level of hierarchy upto which information will be reported.
- If `-depth` is not specified, information is reported for all hierarchical instances.

- When *<depth>* is set to `0` and the `-inst` option is not specified, information is reported for the top module only. When *<depth>* is set to `0` and `-inst` is specified, information specific to *<hier_inst_name>* is reported.

  Note: The `-depth` option cannot be specified with `-mem_list`. However, it can be specified with `-inst`.

-no_wrap

By default, information is printed in fixed -width columns. If information for a field exceeds the column width, information continues on the next line under the same column. This option prevents information in columns from splitting across rows, making it easier to extract information from the report.

-text | -xml | -csv

Specifies the format of the report file to be generated. The information is reported in table format, with a choice of saving the report in text, XML and CSV format. By default, the report is generated as a plain text file.

Note: For CSV files, the semicolon(;) is used as the default delimiter. To specify another delimiter, use the `-csv_delimiter` option.

-csv_delimiter *<csv_delimiter>*

Specifies the delimiter for the CSV report. By default, the semicolon (;) is used as a delimiter.

-filename *<filename>*

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

Related Commands

report_memory_efficiency, report_memory_gating, report_memory_power

Usage

The **report_memory_instances** command reports properties for memories like size, number of read/write ports, power mode and status of the memory enable in the design.

Examples

Example 1: Reports properties for all memories in the design.

```
report_memory_instances
```

Example 2: Reports properties for all memories instantiated within the hierarchical instance `I1.I2`.

```
report_memory_instances  -inst I1.I2
```

Example 3: Reports properties for all memories instantiated in the top module only. The report does not include information for memories within the child instances.

```
report_memory_instances  -depth 0
```

Example 4: Reports properties for the memories `I1.mem1`, `I1.I2.mem`, `mem1`, and `mem2`.

```
report_memory_instances  -mem_list {I1.mem1 I1.I2.mem mem1 mem2}
```

---

report_memory_power

Deprecated w.e.f. PowerPro 10.4. Instead of this command, use report_format, which works in context of the original design. You can also use `report_power -group memory` or `report_format -metric "Memory Info"`.

report_memory_power — Reports static and dynamic power consumption of memories, as computed by PowerPro. Note that this command can only be called after `prototype_design`.

Synopsis

```
report_memory_power
    [ -mem_list <memory_list> |  -inst <hier_inst_name> ]
    [ -depth <depth>] [ -no_wrap] [ -text |  -xml |  -csv ]
    [ -csv_delimiter <csv_delimiter>] [ -filename <filename> ]
```

-mem_list *`<memory_list>`* | -inst *`<hier_inst_name>`*

Specifies the scope of the report.

Specify `-mem_list` to report information only for the memories specified by *`<memory_list>`*. All other options defining the scope of the report are ignored when this option is specified. *Note that the -depth option cannot be specified with -mem_list.*

Specify `-inst` to report information only for memories within the hierarchical instance *`<hier_inst_name>`*. If not specified, the scope is set to the top module.

-depth *`<depth>`*

Specifies the depth of the instance to be reported. By default, information is reported for all levels. When this option is specified, the following rules apply:

- `<depth>` takes the values {0, 1, 2,....}

- If `-inst` is specified, *`<depth>`* indicates the level of hierarchy upto which information will be reported.
- If `-depth` is not specified, power consumption is reported for all hierarchical instances.
- When *`<depth>`* is set to 0 and the `-inst` option is not specified, information is reported for the top module only. When *`<depth>`* is set to 0 and `-inst` is specified, information specific to *`<hier_inst_name>`* is reported.

Note: The `-depth` option cannot be specified with `-mem_list`. However, it can be specified with `-inst`.

-no_wrap

By default, information is printed in fixed-width columns. If information for a field exceeds the column width, information continues on the next line under the same column. This option prevents information in columns from splitting across rows, making it easier to extract information from the report.

-text | -xml | -csv

Specifies the format of the report file to be generated. The information is reported in table format, with a choice of saving the report in text, XML and CSV format. By default, the report is generated as a plain text file.

Note: For CSV files, the semicolon(;) is used as the default delimiter. To specify another delimiter, use the `-csv_delimiter` option.

-csv_delimiter *`<csv_delimiter>`*

Specifies the delimiter for the CSV report. By default, the semicolon (;) is used as a delimiter.

-filename *`<filename>`*

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

Related Commands

report_memory_instances, report_memory_efficiency, report_memory_gating

Usage

The **report_memory_power** command reports dynamic and static power for memories, as computed by PowerPro.

Examples

Example 1: Reports power consumption numbers for all memories in the design.

```
report_memory_power
```

Example 2: Reports power consumption numbers for all memories within the hierarchical instance `I1.I2`.

```
report_memory_power  -inst I1.I2
```

Example 3: Reports power consumption numbers for all memories in the top module only. Power consumption for memories within child instances are not reported.

```
report_memory_power  -depth 0
```

Example 4: Reports power consumption only for the memories `I1.mem1`, `I1.I2.mem`, `mem1`, and `mem2`.

```
report_memory_power  -mem_list {I1.mem1 I1.I2.mem mem1 mem2}
```

report_memory_redundant_access

Deprecated w.e.f. PowerPro 10.4. Many redundancies are available as Metric now.

report_memory_redundant_access — Reports statistics about redundant access opportunities for memories which were not enabled by PowerPro. Redundant memory access information is taken from the FSDB file. Note that the `analyze_mem_sim_traces` command must be called before issuing this command.

Synopsis

```
report_memory_redundant_access [ -summary | -detail ]
    [ -observability ] [ -sleep ] [ -skid_rd ] [ -skid_wr ]
    [ -depth <depth>  [ -mem_list <mem_list> ]
    [ -filter [ clk_domain | no_predicted_val | constraints ] ]
    [ -no_wrap] [ -text | -xml | -csv ]
    [ -csv_delimiter <csv_delimiter> ] [ -filename <filename> ]
```

-summary | -detail

Specifies the amount of information to be reported.

Specify `-summary` to report a summary of potential redundant access opportunities in the design.

Specify `-detail` to report detailed information for memory enables that have potential redundant access opportunities. Information like current and potential dynamic and leakage power numbers, current efficiency numbers and potential efficiency numbers are reported.

-observability

Specifies that the report include information for those memories which have a potential observability based memory redundant access opportunity.

-stability

This option has been deprecated w.e.f PowerPro 7.1. Stability based sequential analysis to identify redundant reads on memories is now performed by default during Cycle Based Redundancy Analysis. The report for memories with potential stability based memory redundant access opportunities is displayed by default in PowerPro Analyzer. A text report is also generated post `prototype_design` and can be viewed on the command prompt. For more information on the reports generated during Cycle Based Redundancy Analysis, refer to the PowerPro User Manual.

-sleep

Specifies that the report include information for those memories which have a potential sleep gating memory redundant access opportunity.

-skid_rd

Specifies that the report include information for those memories which have potential skid based redundant memory read saving opportunity.

-skid_wr

Specifies that the report include information for those memories which have potential skid based redundant memory write saving opportunity.

-depth *`<depth>`*

Specifies the depth of the memory to be reported. By default, information for all memories is reported. *`<depth>`* takes values {0, 1, 2,....}.

-mem_list *`<mem_list>`*

Specifies that the report include information only for those memories listed by *`<mem_list>`*.

-filter [clk_domain | no_predicted_val | constraints]

Specifies that the report include information for those memories that have redundant access opportunities that match the loss of opportunity specified. By default, the report provides information for all reasons.

-no_wrap

By default, information is printed in fixed-width columns. If information for a field exceeds the column width, information continues on the next line under the same column. This option prevents information in columns from splitting across rows, making it easier to extract information from the report.

-text | -xml | -csv

Specifies the format of the report file to be generated. The information is reported in table format, with a choice of saving the report in text, XML and CSV format. By default, the report is generated as a plain text file.

Note: For CSV files, the semicolon(;) is used as the default delimiter. To specify another delimiter, use the `-csv_delimiter` option.

-csv_delimiter *`<csv_delimiter>`*

Specifies the delimiter for the CSV report. By default, the semicolon (;) is used as a delimiter.

-filename *<filename>*

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

Related Commands

analyze_mem_sim_traces, read_fsdb, report_violating_signals, set_powerpro_mode

Usage

The **report_memory_redundant_access** command reports various statistics like initial efficiency, final efficiency and power saving about the redundant access opportunities for memories that were not enabled by PowerPro. It also specifies why PowerPro could not automatically identify the gating conditions for the memories.

Examples

Example 1: Reports potential redundant access opportunities for all memories in the design.

```
report_memory_redundant_access
```

Example 2: Reports potential redundant access opportunities for all memories instantiated in the top module.

```
report_memory_redundant_access  -depth 0
```

Example 3: Reports potential redundant access opportunities for all memories listed by `-mem_list`

```
report_memory_redundant_access  -mem_list {mem1 mem2 mem3 mem4}
```

report_messages

report_messages — Reports message statistics for a current session.

Synopsis

```
report_messages [ -filename <filename> ]
```

-filename *<filename>*

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

Related Commands

disable_msg, enable_msg

Usage

The `report_messages` command reports the number of times a message was reported in the current session. If no filename is specified, the results are printed to the standard output.

Examples

Example 1: Reports message statistics for the current session and writes the information to a file named `messages_count.txt`.

```
report_messages  -filename messages_count.txt
```

An extract from the resultant report file is shown below.

```
Message Tally:
APP-CCFBD     = 1
APP-CCFRF     = 64
APP-CCPG      = 1
APP-FWR       = 3
CDB-CCEL      = 5
CDB-CCLF      = 34
...
```

report_power

report_power — Reports internal and leakage power for a design. Note that this command can only be called after `prototype_design`.

Synopsis

```
report_power
    [ -net_power | -instance_power | -hierarchical |  -
    detailed_hierarchical | -summary ]
    [ -start_instance <inst_name> ]
    [ -only <instance_or_net_list> ]
    [ -depth <depth> ]
    [ -group [io_pad | memory | black_box | register | combinational |
    clock_network ] ]
    [ -nworst <number>] [-clocks <clock_domain>] [ -uniquify_net] [ -
    propagate { on | off } ]
    [ -no_wrap] [ -text | -xml | -csv ]
    [ -csv_delimiter <csv_delimiter>] [ -filename <filename> ]
    [ -with_clock_and_period {<clk_domain_name>, <new_clk_period>} ]
    [ -power_unit <unit_type> ]
    [ -with_clock_and_frequency <with_clock_and_frequency> ]
    [ -time_based ] [ -generate_slope_rank ]
    [ -window <window_size>] [-delta <delta_size>]
    [ -interval_area <interval_value> | -num_slices <slice_count> ]
    [ -threshold <threshold_value> ]
    [ -power_waveform <filename> ]
    [ -pa_html_report ]
    [ -output_dir <dir_name> ]
    [ -power_rail <power_rail_names> ]
```

-net_power | -instance_power | -hierarchical | -detailed_hierarchical |-summary

Specifies the scope of the report.

Specify `-net_power` to report information for nets only.

Specify `-instance_power` to report power information for instances only.

Specify `-summary` to generate a high-level report for the various power groups. If `-start_instance` is also provided, information is reported for *`<inst_name>`* only.

Specify `-hierarchical` to display power information for each block in hierarchical format. The `-depth` option can be used to limit the number of levels for which information is to be reported. With this option, internal, leakage and total power numbers are reported for each hierarchical block. Hierarchy is shown through indentations.
Note: The *`-nworst`* option is not honored when the *`-hierarchical`* option is specified.

Specify `-detailed_hierarchical` to display power detailed information for each block in hierarchical format. Hierarchies having area more than 10% of total design area will be reported by default. The `-depth` option can be used to limit the number of levels for which information is to be reported. The `-depth` option will be applied on full design and default behavior will be changed. With this option, internal, switching, leakage and total power numbers are reported for each power group of each hierarchical block.
Note: The `-nworst` option is not honored when the `-detailed_hierarchical` option is specified.

-start_instance *`<inst_name>`*

Specifies the name of the hierarchical instance for which information is to be reported. By default, information is reported for all instances.

If specified in revamped PA flow, the CAPP waveform will be generated for the list of specified hierarchies only. By default, waveform will be generated for top hierarchy only. Top hierarchy can be specified as `TOP_HIER` with other hierarchies in the list.

If specified with `-detailed_hierarchical` option, list of hierarchies can be passed.

Note: This option cannot be specified with the `-only option`.

-only

Specifies the list of instances or nets to be displayed with `-net_power` and `-instance_power`.

With this option, only the instances or nets specified by *`<instance_or_net_list>`* are reported.

-depth *`<depth>`*

Specifies the depth of the instance to be reported. By default, information is reported for all levels. This option accepts any positive integer greater than `1`.

-group [io_pad | memory | black_box | register | combinational | clock_network]

Specifies that the report include information for a specific type of object only. The `-group` option can be used along with `-instance_power`, `-hierarchical`, `-detailed_hierarchical`, and `-summary` to further restrict the amount of information reported.

-nworst *<number>*

Specifies that the report include information only for the highest power-consuming objects. Note that this option is honored only when *-net_power* or *-instance_power* is specified.

-clocks <*clock_domain*>

Specifies that the report includes information for the list of clock domains. The `-clocks` option can be used along with `-summary`,`-instance_power`, `-start_instance`, `-net_power` along with `-uniquify_net`, and `-hierarchical` to further restrict the amount of information reported.
Note that this option can be used only if PowerPro is launched using the `-pa` option or in Gate Level Power Analysis Flow. Only user clock domains can be specified with the `-clocks` option. The elements which are not under the user clock domain will be reported under clock-domain `virtual_clock`.

-uniquify_net

If the `-uniquify_net` option is specified with the `-net_power` option, only those nets that are directly connected to their electrical sources are reported. The nets in the fanout of the hierarchical boundaries and the bit steering logic are not reported.

-propagate { on | off }

Forces PowerPro to estimate switching activity for unasserted nets in the design; applicable only when the switching activity annotated from the SAIF files (using the `read_saif` command) is not sufficient.

Default is `on`. When set to `off`, an error will be generated if the SAIF annotation on sequential outputs is less than 90%.

Note that report generation might take a while due to the time taken by switching activity propagation.

-no_wrap

By default, information is printed in fixed-width columns. If information for a field exceeds the column width, information continues on the next line under the same column. This option prevents information in columns from splitting across rows, making it easier to extract information from the report.

-text | -xml | -csv

Specifies the format of the report file to be generated. The information is reported in table format, with a choice of saving the report in text, XML and CSV format. By default, the report is generated as a plain text file.

Note: For CSV files, the semicolon(;) is used as the default delimiter. To specify another delimiter, use the `-csv_delimiter` option.

-csv_delimiter *<csv_delimiter>*

Specifies the delimiter for the CSV report. By default, the semicolon (;) is used as a delimiter.

-filename *<filename>*

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

-with_clock_and_period *<with_clock_and_period>*

Specifies that the power of a design region, that is impacted by single clock domain, is a direct function of clock period. This option gives flexibility to specify a new clock period for a clock domain, based on which, power should be reported.

As an argument to this option, specify a pair like `{clk123 10}` where `clk123` is the clock domain name (specified using the `-name` option while specifying the `create_clock` command), and `10` is the new clock period (applicable to all clock ports under the `clk123` domain).

To override multiple clock domain periods, specify this option multiple times.

Note that this option only affects the output of the current `report_power` command. For design regions influenced by multiple clock domains, the power would be scaled based on faster clock period ratio.

-power_unit <*unit_type*>

Specifies the base power unit. Possible values are `fW`, `pW`, `nW`, `uW`, `mW`, and `W`. Default value is `uW`.

-with_clock_and_frequency <*with_clock_and_frequency*>

Specifies the clock frequency (in MHz) that must be used during power analysis for a given clock domain. This clock frequency can be different from the clock frequency that is annotated using the switching activity file. To override multiple frequencies of the clock domain, specify this option multiple times.

1. This option does not impact the synthesis done by PowerPro.
2. This option affects only the output of the current report_power command.
3. For design regions that are influenced by multiple clock domains, the power is scaled based on the higher clock frequency ratio.

-time_based

Enables the time-based power analysis. If this option is not used, the tool will run in the average-based power analysis mode. The `time_based` option can be used only if the `read_fsdb` command has run successfully.

-generate_slope_rank

The Slope Rank report gives us the slope of values at two consecutive timestamps in time-based power computation. You can generate this report for Cycle Accurate Peak Power (CAPP) and Sliding Window Peak Power (SWPP) plots. In this report, the data is sorted in descending order for positive values and in ascending order for negative values.

Note: The slope rank report will be generated only if the `–generate_slope_rank` option is specified with the `report_power -time_based` command only in the Enhanced-PA flow.

-window <*window_size*>

Specifies the size of the `window` used for generating sliding window peak power waveform.
Note: This option is supported in the enhanced PA flow only when the `-time_based` option has been used.

-delta <*delta_size*>

Specifies the size by which the sliding window moves for generating sliding window peak power waveform.
Note: This option is supported in the enhanced PA flow only when the `-window` option has been used.

-interval_area *<interval_value>* | -num_slices *<slice_count>*

Divides the total simulation time into multiple intervals of size `interval_value` to invoke event-by-event power analysis. If the time unit is not specified, PowerPro automatically infers it from the FSDB file. Supported time units are: `s`, `ms`, `us`, `ns`, `ps`, `fs`, `as`, `zs`, and `ys`. The variable accepts only floating point and time values.

`interval_area` can be used only with the `time_based` option.

`num_slices` divides the total simulation time into specified `slice_count` intervals. The size of the interval will be calculated according to the provided `slice_count`.

`num_slices` can be used only with the `time_based` option.

-threshold <*threshold_value*>

If power variation of two consecutive time windows is greater than or equal to `threshold_value`, the later time-window is marked as high di/dt (i.e. rate of change of current from power supply) region. Supported power units are: `W`, `mW`, `uW`, `nW`, `pW`, `fW`, `aW`, `zW`, and `yW`. Default power unit is uW.

1. The `threshold` option can be used with the `interval_area | num_slices` option only under the `time_based` option in Gate Level power analysis flow.

2. If the `threshold` option is not specified, then the largest power variation between any two consecutive time-windows is assumed as the `threshold_value`.

-power_waveform <*filename*>

Dumps a `vcd` file that contains the time-ordered power number changes for different power groups. This option can be used only if the `time_based` option has been used.

If specified in enhanced PA flow, dumps vcd files for CAPP waveform.

-pa_html_report

Generates HTML page for each design hierarchy which contains power summary report and power information for all instances in the design hierarchy. All files are created in a directory specified using `-output_dir` option. Only options `–output_dir` and `-depth` are supported with this option. Name of the starting HTML page is `top.html`

-output_dir <*dir_name*>

Specifies the directory in which HTML pages for design hierarchies are to be generated. This option can only be used with `-pa_html_report`. By default all HTML pages are generated in the directory named `pa_html_report` in the work directory for the run.

-power_rail <*power_rail_names*>

Specifies whether to report power for the design objects that are connected to the specific set of power rails. This option will work only when Unified Power Format (UPF) has been read in the flow and the corresponding power rails have been defined as Supply Nets in the UPF.

Power-rail-wise reporting is currently available only for Average Power Reporting Flow.

Note that the '/' key must be used as the hierarchy delimiter, if a hierarchical name is specified using the `-power_rail` option.

Related Commands

set_powerpro_mode, prototype_design

Usage

The **report_power** command reports power numbers for a design, based on switching activity. If switching activity has not been annotated using the `read_saif` command or, if not all nets are asserted, then in presence `-propagate on` option it will estimate the switching activity of unasserted nets in the design based on the switching activity of the nets in their transitive fanin cones. If primary inputs have no switching activity asserted then they default to a toggle rate of 0.005/ns and a probability of 0.5.

Examples

Example 1: Reports summary of power for the design, along with the environment setup information.

```
report_power  -summary
```

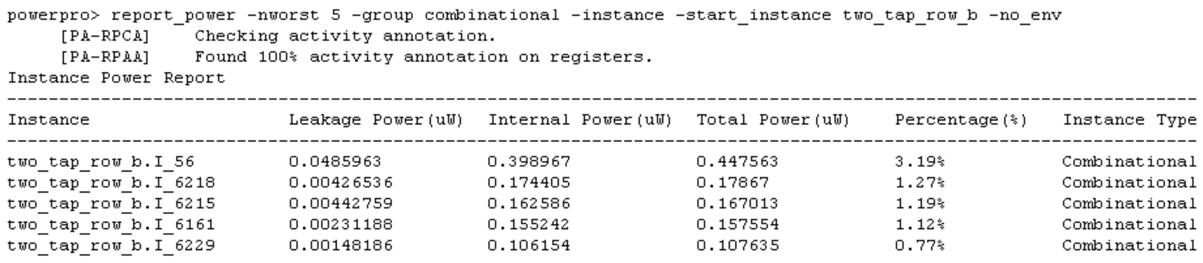
A sample report is shown below.

```
Library(s) Used:
  ff_lv32_0c
  ff_lv32_0ch


Operating Conditions    : ff_lv32_0c
Wireload Mode           : none
  ----------------------------------------------------------------------------------------------------------------
  Design                    | Target Libraries| Voltage | Wireload Model   | Library for Wireload | VTH Distribution
  ----------------------------------------------------------------------------------------------------------------
  correlator_top             All              1.32     tsmc13_wl10       ff_lv32_0c
  ecc_inst.ecc               All              1.32     tsmc13_wl10       ff_lv32_0c             lvth:100.00%
  count_inst.count_start     All              1.32     tsmc13_wl10       ff_lv32_0c             hvth:22.00%,lvth:78.00%
  correlate_rows.correlator  All              1.32     tsmc13_wl10       ff_lv32_0c             hvth:20.34%,lvth:79.66%
  two_tap_row_a.two_tap      All              1.32     tsmc13_wl10       ff_lv32_0c             hvth:20.00%,lvth:80.00%
  two_tap_row_b.two_tap      All              1.32     tsmc13_wl10       ff_lv32_0c             hvth:20.00%,lvth:80.00%
  decode_row.decode          All              1.32     tsmc13_wl10       ff_lv32_0c             hvth:20.00%,lvth:80.00%

  ----------------------------------------------------------------------------------------------------------------

Global Operating Voltage = 1.32
Power-specific unit information:
  Voltage Units       = 1V
  Capacitance Units   = 1pf
  Time Units          = 1ns
  Dynamic Power Units = 1uW
  Leakage Power Units = 1uW


Power Summary Report
------------------------------------------------------------------------------------------------------
Power Group    Count          Leakage Power(uW)   Internal Power(uW)   Total Power(uW)     Percentage(%)
------------------------------------------------------------------------------------------------------
io_pad         0              0                   0                    0                   0%
memory         0              0                   0                    0                   0%
black_box      0              0                   0                    0                   0%
register       200            9.48018             42.8547              52.3349             37.98%
combinational  427            4.33971             34.9362              39.2759             28.51%
sequential     8              0.214317            2.38012              2.59444             1.88%
clock_network  1              0.125793            43.4479              43.5736             31.63%
total          636            14.16               123.619              137.779             100%
```

Example 2: Prints a summary for the group 'register' (envirnonment setup information is suppressed).

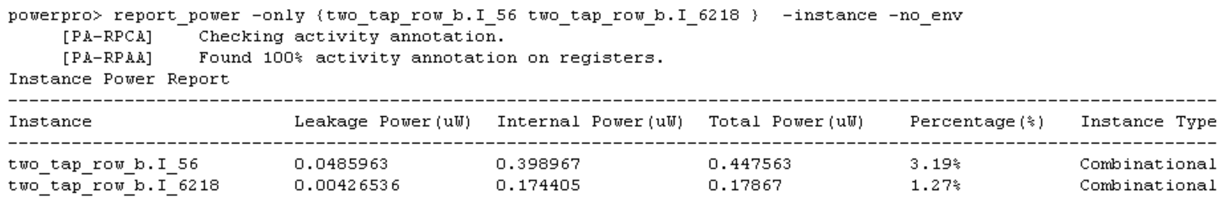
```
powerpro> report_power -summary -group register -no_env
     [PA-RPCA]    Checking activity annotation.
     [PA-RPAA]    Found 100% activity annotation on registers.
Power Summary Report
------------------------------------------------------------------------------------------------------
Power Group    Count          Leakage Power(uW)   Internal Power(uW)   Total Power(uW)     Percentage(%)
------------------------------------------------------------------------------------------------------
register       200            1.32077             121.105              122.426             77.41%
```

Example 3: Generates a report for the top 5 combinational instances consuming maximum power that are inside the hierarchical instance 'two_tap_row_b' (environment setup information is suppressed).



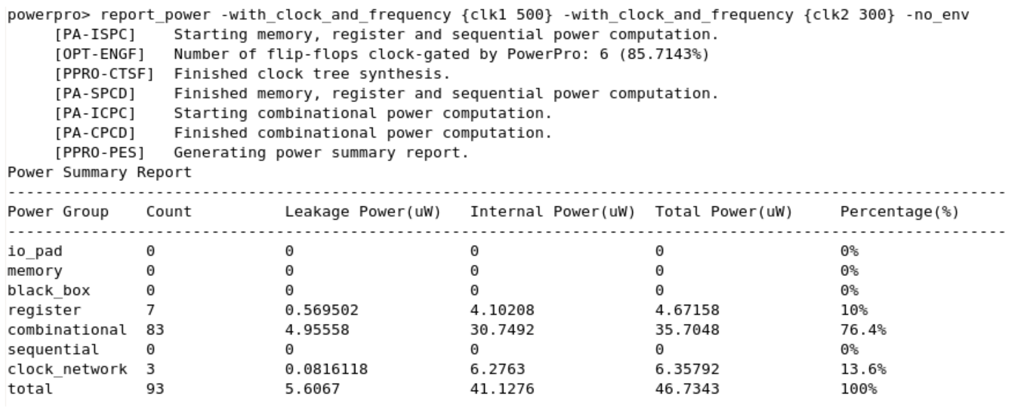
```
powerpro> report_power -nworst 5 -group combinational -instance -start_instance two_tap_row_b -no_env
     [PA-RPCA]    Checking activity annotation.
     [PA-RPAA]    Found 100% activity annotation on registers.
Instance Power Report
----------------------------------------------------------------------------------------------------------
Instance                    Leakage Power(uW)  Internal Power(uW)  Total Power(uW)    Percentage(%)   Instance Type
----------------------------------------------------------------------------------------------------------
two_tap_row_b.I_56          0.0485963          0.398967            0.447563           3.19%           Combinational
two_tap_row_b.I_6218        0.00426536         0.174405            0.17867            1.27%           Combinational
two_tap_row_b.I_6215        0.00442759         0.162586            0.167013           1.19%           Combinational
two_tap_row_b.I_6161        0.00231188         0.155242            0.157554           1.12%           Combinational
two_tap_row_b.I_6229        0.00148186         0.106154            0.107635           0.77%           Combinational
```

Example 4: Generates a report for specific instance (environment setup information is suppressed).



```
powerpro> report_power -only {two_tap_row_b.I_56 two_tap_row_b.I_6218 }  -instance -no_env
     [PA-RPCA]    Checking activity annotation.
     [PA-RPAA]    Found 100% activity annotation on registers.
Instance Power Report
----------------------------------------------------------------------------------------------------------
Instance                    Leakage Power(uW)  Internal Power(uW)  Total Power(uW)    Percentage(%)   Instance Type
----------------------------------------------------------------------------------------------------------
two_tap_row_b.I_56          0.0485963          0.398967            0.447563           3.19%           Combinational
two_tap_row_b.I_6218        0.00426536         0.174405            0.17867            1.27%           Combinational
```

Example 5: Uses the new frequencies `500 MHz` and `300 MHz` for the domains `clk1` and `clk2` respectively.



```
powerpro> report_power -with_clock_and_frequency {clk1 500} -with_clock_and_frequency {clk2 300} -no_env
    [PA-ISPC]    Starting memory, register and sequential power computation.
    [OPT-ENGF]   Number of flip-flops clock-gated by PowerPro: 6 (85.7143%)
    [PPRO-CTSF]  Finished clock tree synthesis.
    [PA-SPCD]    Finished memory, register and sequential power computation.
    [PA-ICPC]    Starting combinational power computation.
    [PA-CPCD]    Finished combinational power computation.
    [PPRO-PES]   Generating power summary report.
Power Summary Report
---------------------------------------------------------------------------------------------------------
Power Group    Count         Leakage Power(uW)  Internal Power(uW)  Total Power(uW)     Percentage(%)
---------------------------------------------------------------------------------------------------------
io_pad         0             0                  0                   0                   0%
memory         0             0                  0                   0                   0%
black_box      0             0                  0                   0                   0%
register       7             0.569502           4.10208             4.67158             10%
combinational  83            4.95558            30.7492             35.7048             76.4%
sequential     0             0                  0                   0                   0%
clock_network  3             0.0816118          6.2763              6.35792             13.6%
total          93            5.6067             41.1276             46.7343             100%
```

Example 6: Generates HTML power report for all design hierarchies in the directory HTML_PA_REPORT.

```
powerpro> report_power -pa_html_report -output_dir HTML_PA_REPORT
```

Example 7: Generates separate power reports for power rails `vddcx` and `vddmx`.

```
powerpro> report_power -power_rail vddcx -filename top_power_vddcx.rpt
powerpro> report_power -power_rail vddmx -filename top_power_vddmx.rpt
```

Example 8: Generates hierarchical power report for power rails for a lower hierarchy.

```
powerpro> report_power -hier -power_rail top/mid/vddcx -filename
hier_power_vddcx.rpt
```

report_qor

report_qor — Reports CG sequential optimization results. Note that this command can only be called after `prototype_design`.

Synopsis

```
report_qor
    [ -observability | -stability | -stability_c | -stability_s ]
    [ -reset_pin_added ] [ -synchronizers_added ] [ -level <level> ]
    [ -text | -xls | -csv ] [ -csv_delimiter <csv_delimiter> ]
    [ -filename <filename> ]
```

-observability | -stability | -stability_c | -stability_s

Specify `-observability` to report results of the observability-based CG sequential optimization.

Specify `-stability` to report results of the stability-based CG sequential optimization. Report will include results for both constant and symbolic stability-based optimization.

Specify `-stability_c` to report results of the constant stability-based CG sequential optimization.

Specify `-stability_s` to report results of the symbolic stability-based CG sequential optimization.

-reset_pin_added

Specifies that information about PowerPro-added reset ports be included in the report; this includes information like the name and polarity of the reset port.

-synchronizers_added

Specifies that information about PowerPro-added synchronizers be included in the report.

-level *<level>*

Specifies the amount of information to be printed. This controls the number of levels within the hierarchy for which information is to be reported. Default is `2`. *Note that this only works with the `-xls` option.*

-text | -xls | -csv

Specifies the format of the report file to be generated. The information is reported in table format, with a choice of saving the report in text, XML and CSV format. By default, the report is generated as a plain text file.

Notes: For CSV files, the semicolon(;) is used as the default delimiter. To specify another delimiter, use the `-csv_delimiter` option.

-csv_delimiter *<csv_delimiter>*

Specifies the delimiter for the CSV report. By default, the semicolon (;) is used as a delimiter.

-filename *<filename>*

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

Related Commands

insert_observability_logic, insert_stability_logic, prototype_design

Usage

The **report_qor** command reports a summary of CG sequential optimization results. The report includes information about Moves, CG Efficiency after optimization, Register power saving, Area change, details of PowerPro enabled flops and, information about synchronizers and reset ports added by PowerPro.

Examples

Example 1: Reports CG sequential optimization results for all instances in the design.

```
report_qor
```

Example 2: Reports observability-based CG sequential optimization results for all instances in the design.

```
report_qor -observability
```

Example 3: Reports constant stability-based CG sequential optimization results for all instances in the design.

```
report_qor -stability_c
```

Example 4: Reports stability-based CG sequential optimization results for all instances in the design and writes the output to an XLS file named `qor_rpt.xls`.

```
report_qor -stability -xls -filename qor_rpt.xls
```

Example 5: Reports CG sequential optimization results for all flops upto the second level hierarchy of the top module and writes the output to an XLS file named `qor_rpt.xls`.

```
report_qor -level 2 -xls -filename qor_rpt.xls
```

report_redundant_mux_activity

Deprecated w.e.f. PowerPro 10.4. Instead of this command, use `report_format -metric "Redundant Mux Toggles"`.

report_redundant_mux_activity — Reports information about redundant toggle activity for muxes in the design. Note that this command can only be called after `prototype_design` if the `generate_guidance` command has been run in the flow.

Synopsis

```
report_redundant_reset_flops [ -no_wrap] [ -text | -csv ]
    [ -csv_delimiter <csv_delimiter> ] [ -filename <filename> ]
```

-no_wrap

By default, information is printed in fixed-width columns. If information for a field exceeds the column width, information continues on the next line under the same column. This option prevents information in columns from splitting across rows, making it easier to extract information from the report.

-text | -csv

Specifies the format of the report file to be generated. The information is reported in table format, with a choice of saving the report in text and CSV format. By default, the report is generated as a plain text file.

Note: For CSV files, the semicolon(;) is used as the default delimiter. To specify another delimiter, use the `-csv_delimiter` option.

-csv_delimiter *<csv_delimiter>*

Specifies the delimiter for the CSV report. By default, the semicolon (;) is used as a delimiter.

-filename *<filename>*

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

Related Commands

report_high_toggling_signals

Usage

The **report_redundant_mux_activity** command reports percentage redundant toggle activity on the inputs of the muxes. Additionally, it reports percentage wasted toggles, width of input signal, number of fanin of the input signal and source relation with line number for these muxes.

Note: The numbers reported by the **report_redundant_mux_activity** command do not get updated post sequential optimization. It is recommended that this command be called before any sequential optimization command. This limitation will be fixed in a future release.

Examples

Example 1: Generates a report with redundant mux activity and writes the output to a text file named `rrma_rep.rpt`.

```
report_redundant_mux_activity -filename rrma_rep.rpt
```

Example 2: Generates a report with redundant mux activity and writes the output to a CSV file named `rrma_exp.csv`. The comma(,) is used as a delimiter.

```
report_redundant_mux_activity -filename rrma_exp.csv -csv -
csv_delimiter ,
```

report_redundant_reset_flops

Deprecated w.e.f. PowerPro 10.4. Instead of this command, use `report_format –metric "Redundant Reset"`.

report_redundant_reset_flops — Reports flops with redundant resets. Note that this command can only be called after `insert_observability_logic`.

Synopsis

```
report_redundant_reset_flops [ -filename <filename> ]
    [ -no_wrap] [ -text | -csv ]
    [ -csv_delimiter <csv_delimiter> ]
```

-filename *<filename>*

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

-no_wrap

By default, information is printed in fixed -width columns. If information for a field exceeds the column width, information continues on the next line under the same column. This option prevents information in columns from splitting across rows, making it easier to extract information from the report.

-text | -csv

Specifies the format of the report file to be generated. The information is reported in table format, with a choice of saving the report in text and CSV format. By default, the report is generated as a plain text file.

Note: For CSV files, the semicolon(;) is used as the default delimiter. To specify another delimiter, use the `-csv_delimiter` option.

-csv_delimiter *<csv_delimiter>*

Specifies the delimiter for the CSV report. By default, the semicolon (;) is used as a delimiter.

Related Commands

insert_observability_logic

Usage

The `report_redundant_reset_flops` command reports those flops with redundant resets. The report also includes information about potential power savings as a result of removing the reset on the flop.

Examples

Example 1: Generates a report detailing the flops with redundant resets in the design. The report is generated in text format and written to the `redundant_reset_flops.rpt` file.

```
report_redundant_reset_flops  -filename redundant_reset_flops.rpt
```

A sample report is shown below.

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - - -
Domain Flop Object  Width  Reset Signal  Polarity   Reset Type
Potential Power Saving(uW)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
- -- -
1      pff1         8       rst            POS        ASYNC      10.0134
2      pff2         4       rst            NEG        ASYNC      5.764
```

Example 2: Generate a report detailing the flops with redundant resets and writes the output to the `redundant_reset_flops.rpt` file with the colon(:) as the delimiter.

```
report_redundant_reset_flops   -filename redundant_reset_flops.rpt
-csv  -csv_delimiter :
```

report_redundant_write_info

Deprecated w.e.f. PowerPro 10.4. Instead of this command, use `report_format -metric "Clock Toggle - Data Stable"`.

report_redundant_write_info — Reports information about potential redundant writes in the design. The report also specifies why PowerPro could not automatically identify gating conditions to the flops. Note that this command can only be called after the `generate_guidance` and `insert_observability_logic` commands have been run in the flow. When this command is called with the `-stability` option, the `report_redundant_write_info` command can be called before or after `insert_observability_logic`.

Synopsis

```
report_redundant_write_info [ -summary | -detail ]
    [ -observability |  -stability ] [ -depth <depth> ]
    [ -ff_list <flop_list> ]
    [ -filter [ clk_domain | no_predicted_val | constraints ] ]
    [ -flop_width <flop_width> ] [ -no_wrap] [ -text |  -xml |  -csv ]
    [ -csv_delimiter <csv_delimiter> ] [ -filename <filename> ]
```

-summary | -detail

Specifies the amount of information to be reported.

Specify `-summary` to report a summary of flops with potential redundant write opportunities. The report includes information like register power saving and efficiency change for such flops.

Specify `-detail` to report detailed information for flops with potential redundant write opportunities. The report includes information like the reason PowerPro was unable to identify the gating condition to the flop, initial efficiency, final efficiency and register power saving.

-observability | -stability

Specifies the type of redundant write to be reported.

Specify `-observability` to report information for flops with a potential of observability based redundant write.

Specify `-stability` to report information for flops with a potential of stability based redundant write.

-depth *<depth>*

Specifies the depth of the flops to be reported. If this option is not specified, information is reported for all flops in the design. Legal values are {0, 1, 2,....}.

-ff_list *<flop_list>*

Specifies that data be written only for the flops listed by *<flop_list>*.

-filter [clk_domain | no_predicted_val | constraints]

Specifies the reason to be reported. Use this option to filter the report to include information specific to a type of loss of opportunity. By default, the report includes data for all reasons.

Specify `clk_domain` to report those flops which were not gated by PowerPro because the input for the control logic was not in the same clock domain as the flop being gated.

Specify `no_predicted_val` to report those flops which were not automatically gated because PowerPro could not determine the previous cycle value of the corresponding control signal. This could be because the signals were being driven by a primary input, black box output or hard -operator.

Specify `constraints` to report those flops which were not gated by PowerPro because a constraint like `set_ignore_signal` was applied.

-flop_width *<flop_width>*

Specifies the width of flops for which information is to be reported. Default is `4`.

-no_wrap

By default, information is printed in fixed -width columns. If information for a field exceeds the column width, information continues on the next line under the same column. This option prevents information in columns from splitting across rows, making it easier to extract information from the report.

-text | -xml | -csv

Specifies the format of the report file to be generated. The information is reported in table format, with a choice of saving the report in text, XML and CSV format. By default, the report is generated as a plain text file.

Note: For CSV files, the semicolon(;) is used as the default delimiter. To specify another delimiter, use the `-csv_delimiter` option.

-csv_delimiter *<csv_delimiter>*

Specifies the delimiter for the CSV report. By default, the semicolon (;) is used as a delimiter.

-filename *<filename>*

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

Related Commands

report_violating_signals, set_powerpro_mode

Usage

The **report_redundant_write_info** command reports various statistics like initial efficiency, final efficiency and power saving for flops that could not be gated by PowerPro. It also specifies why PowerPro could not automatically identify the gating conditions to the flops.

Examples

Example 1: Reports information about potential redundant writes in the design.

```
report_redundant_write_info
```

Example 2: Reports information about potential redundant writes in the top module only. Information specific to its child instances is not reported.

```
report_redundant_write_info  -depth 0
```

Example 3: Reports information about potential redundant writes specific to the flops `f1`, `f2`, `f3` and `f4` only.

```
report_redundant_write_info  -ff_list {f1 f2 f3 f4}
```

report_run_summary

report_run_summary — Fetches all the important and actionable warning messages from various logs and stores them when the command has run successfully. Note that this command can only be called after the `prototype_design` command.

Synopsis

`report_run_summary` -unroll <*warning_unrolling_depth*>

-unroll <*warning_unrolling_depth*>

Specifies the level of unrolling for warnings. It is used to unroll the similar warnings for multiple objects to the specified level in the same command run. Legal values include any positive integer. Default is `1`.

Related Commands

report_format

Usage

The report_run_summary command collects DICs (Data Integrity Checks) for PowerPro run in `<<`*`work_dir`*`>>/dic.log`. When the command has run successfully, a comprehensive report *<<work_dir>>*/run_summary.log is generated that prints all the important warning messages and information.

Examples

Example 1: Returns run summary for the complete powerpro run with warnings expanded upto 4 (four) levels.

```
read_library techlib.lib
build_design input.v
read_saif input.v.saif
prototype_design
report_run_summary -unroll 4
```

Example 2: Returns run summary for the complete powerpro run with warnings expanded upto default level 1.

```
read_library techlib.lib
build_design input.v
read_saif input.v.saif
prototype_design
report_run_summary
```

---

report_static_signal_gating

Deprecated w.e.f. PowerPro 10.4.

report_static_signal_gating — Reports gating opportunities identified during Static Signal-Based Gating. Note that this command can only be called after insert_static_signal_gating if the generate_guidance command has been run in the flow.

Synopsis

```
report_static_signal_gating [ -filename <filename> ] [ -no_wrap ]
    [ -text |  -xml |  -csv ] [ -csv_delimiter <csv_delimiter> ]
```

-filename *<filename>*

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

-no_wrap

By default, information is printed in fixed -width columns. If information for a field exceeds the column width, information continues on the next line under the same column. This option prevents information in columns from splitting across rows, making it easier to extract information from the report.

-text | -xml | -csv

Specifies the format of the report file to be generated. The information is reported in table format, with a choice of saving the report in text, XML and CSV format. By default, the report is generated as a plain text file.

Note: For CSV files, the semicolon(;) is used as the default delimiter. To specify another delimiter, use the `-csv_delimiter` option.

-csv_delimiter *<csv_delimiter>*

Specifies the delimiter for the CSV report. By default, the semicolon (;) is used as a delimiter.

Related Commands

insert_static_signal_gating, report_enable_expression, set_static_signal

Usage

The `report_static_signal_gating` command reports gating opportunities identified in the design. Additionally, it reports power changes, efficiency changes and area changes for the gating opportunities identified during Static Signal-Based Gating.

Examples

Example 1: Generates the Static Signal-Based Gating report in text format and writes it to the `rdg_rep.rpt` file.

```
report_static_signal_gating  -filename rdg_rep.rpt
```

Example 2: Generates the Static Signal-Based Gating report and writes the output to the `rdg_exp.csv` file with the colon(:) as the delimiter.

```
report_static_signal_gating  -filename rdg_exp.csv  -csv  \
      -csv_delimiter :
```

report_toggle_profile

report_toggle_profile —Analyses the toggles in the design signals for the FSDB file or the QWave file to generate activity data. The command can be run only after the `read_fsdb` or the `read_qwave` command.

Synopsis

```
report_toggle_profile
    [-window <window_size>] [-delta <delta_size>]
    [-start_instance <inst_name>] [-toggle_waveform <file_name>]
    [-nworst <number>] [ -text | -csv | -csv_delimiter <delimiter>]
```

-window <*window_size*>

If specified, this will enable generation of Moving average waveform over toggle profile data with the given window size.

-delta `<delta_size>`

Specifies the size by which the average window moves for generating Moving Average plot.

-start_instance <*inst_name*>

If specified, all reports and waveforms will be generated for the specified hierarchies only.
Note: Default behaviour will be similar to the time_based flow of `report_power` in terms of hierarchies reported. When specifying list of hierarchies, top hierarchy can be specified as `TOP_HIER` with other hierarchies in the list.

-toggle_waveform <*file_name*>

Specifies the file path where VCD information will be dumped.

-nworst <`number`>

Specifies maximum number of rows that can be written in text reports. Default will be `1000`.

-text | -csv | -csv_delimiter <*delimiter*>

Use the `-text` option to generate the reports in the text format.

Use the `-csv` option to generate the reports in the csv format.

Use the additional option, `-csv_delimiter` with the `-csv` option to specify a custom delimiter to be used in the report csv file.

Note: The options `-text` and `-csv` are mutually exclusive.

Related Commands

get_sa, read_fsdb, read_saif, read_qwave

Usage

The **report_toggle_profile** command reports the toggle activity for a design, using the design signals from the given FSDB file or QWave file. This command can generate text reports as well as plot information, which can be viewed and inspected using GUI. Reports and plots can be generated for any hierarchy at timestamp level granularity.

Examples

Example 1: Generating reports for the top hierarchical instance.

```
report_toggle_profile
```

Note: The location of the text reports will be printed on screen.
Note: Plot information will automatically be available in GUI.

Example 2: Generating reports for multiple instances.

```
report_toggle_profile -start_instance {inst1.inst1_1 inst1.inst1_2
inst2 inst3}
```

Note: To specify the top hierarchy explicitly, use `TOP_HIER` or `TOP_NODE`.

Example 3: Generating moving average reports.

```
report_toggle_profile -window 10000 -delta 2000
```

Note: `window` must always be an integral multiple of `delta`.

Example 4: Generating VCD file.

```
report_toggle_profile -toggle_waveform <path>
```

Example 5: Limiting number of rows in text reports.

```
report_toggle_profile -nworst 500
```

Note: The default value of `nworst` is `1000`.

Example 6: Generating reports in CSV format.

```
report_toggle_profile -csv
report_toggle_profile -csv -csv_delimiter ,
```

Note: Default value of `-csv_delimiter` is `;` (`semi-colon`).

report_violating_signals

Deprecated w.e.f. PowerPro 10.4.

report_violating_signals — Reports signals that have prevented an observability based clock -gating for memories and flops in the design. This command also reports the list of domain-ids, as reported by the `report_redundant_write_info` and `report_memory_redundant_access` command; potential power savings for these signals and reasons for loss of gating opportunity are also reported. Note that this command can only be called after `insert_observability_logic` or `insert_mem_observability_gating` if the `generate_guidance` command has been run in the flow.

Synopsis

```
report_violating_signals
    [ -filter [ clk_domain | no_predicted_val | constraints ] ]
    [ -flop_width <flop_width> ] [ -no_wrap] [ -text |  -xml |  -csv ]
    [ -csv_delimiter <csv_delimiter> ] [ -filename <filename> ]
```

-filter [clk_domain | no_predicted_val | constraints]

Specifies the type of reason (loss of opportunity) for which information is to be reported. By default, information is shown for all violating signals, irrespective of the reason.

-flop_width *<flop_width>*

Specifies the minimum width of flops to be reported. Flops whose width is lesser than the specified value are excluded from the report. Default width is `4`.

-no_wrap

By default, information is printed in fixed -width columns. If information for a field exceeds the column width, information continues on the next line under the same column. This option prevents information in columns from splitting across rows, making it easier to extract information from the report.

-text | -xml | -csv

Specifies the format of the report file to be generated. The information is reported in table format, with a choice of saving the report in text, XML and CSV format. By default, the report is generated as a plain text file.

Note: For CSV files, the semicolon(;) is used as the default delimiter. To specify another delimiter, use the `-csv_delimiter` option.

-csv_delimiter *<csv_delimiter>*

Specifies the delimiter for the CSV report. By default, the semicolon (;) is used as a delimiter.

-filename *<filename>*

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output.

Related Commands

report_redundant_write_info, report_memory_redundant_access, set_powerpro_mode

Usage

The **report_violating_signals** command reports information similar to `report_redundant_write_info` and `report_memory_redundant_access` except that the report is generated based on the reason for violation.

Examples

Example 1: Generates a report listing violating signals and writes the report to the text file `rvs_rep.rpt`.

```
report_violating_signals  -filename rvs_rep.rpt
```

selective_optimization

selective_optimization — Provides capability to select or de-select sequential optimization opportunities across flop(s) instances from a given PowerPro-optimized RTL. The output of this command is a new PowerPro-optimized RTL, optimized DB, and SlecPro-based verification setup to validate this RTL against the non-optimized RTL. User can have multiple variants of PowerPro optimized RTL and verification setup based on filter options available with this command. By default, selective optimization command is run in parallel sub-sessions.

Synopsis

```
selective_optimization {-path <original_run_directory_path>}
[-filter_inst <instance_list>] [-opt_file <opt_file_path>]
[-filter_file <filter_files_path_list>] [-output_dir
    <new_patched_rtl_path>]
```

`-path` *<original_run_directory_path>*

Specifies the working directory of an existent PowerPro-optimized run. This option enables the command to be called at the new PowerPro prompt.

`-filter_inst` *<instance_list>*

Specifies the list of flop(s) instances on which sequential optimization needs to be de-selected.

`-opt_file` *<opt_file_path>*

Specifies the opt file path containing optimization constraints such as `set_observable` and `set_override_stability`.

`-filter_file` *<filter_files_path_list>*

Specifies the list of files containing instances along with their optimization type to remove clock gating opportunities on them. For filter file generation, use the `Clock Gating Report` or the `report_format` command. We can select/de-select clock gating opportunities from an existing visualizer view of Clock Gating report. The file generated can be passed to this option. Separate subsession will run for every file provided with this option. For example, to generate a list of flops with efficiency less than `10%`:

```
report_format -metric "Clock Gating" -filter_column {("Efficiency
Change(%)" <= "10")}  -column_list {FlipFlop,TYPE} -configuration
"DEFAULT" -filename filter_file -csv
```

`-output_dir` *`<new_patched_rtl_path>`*

> Specifies the name of output directory where the selective optimized PowerPro RTL, optimized DB, and verification setup support files are to be stored. Default is current working directory.

Related Commands

set_observable, set_override_stability, report_format

Usage

This command can be invoked in an new shell or an existing shell having valid clock gating opportunities. For each filter criteria, a new light-weight PowerPro session would be executed to generate new PowerPro optimized RTL, optimized DB, and verification setup. This command supports capability to filter out sequential optimization across instances for which SlecPro hard maps are generated. To run parallel sub-sessions, user needs to provide settings for `max_local_process_usage`, `max_remote_process_usage`, `ppro_machine_list_file`, and `host_setup_configuration`.

Examples

Example 1: De-selecting sequential optimization across a list of flop instances.

```
set_global max_local_process_usage 1
selective_optimization -filter_inst {out hier.out1 hier.out2} -path
calypto -output_dir selopt
```

Example 2: De-selecting sequential optimization across instances provided by SlecPro hard maps and instances present in `filterfile1` and `filterfile2`.

```
set_global max_local_process_usage 3
selective_optimization -opt_file powerpro_hard_maps.tcl -filter_file
{filterfile1 filterfile2} -path calypto
```

set_annotated_power

set_annotated_power — Sets the power consumption for blackboxed instances. Note that this command can only be called between `build_design` and `prototype_design`.

Synopsis

```
set_annotated_power -leakage_power <number> -internal_power <number>
    -inst_list <inst_list> [ -unit_base <fW | pW | nW | uW | mW | W> ]
```

-leakage_power *`<number>`*

Specifies the leakage power consumption of black box instances in the existing power unit.

-internal_power *<number>*

Specifies the internal power consumption of black box instances in the existing power unit.

-inst_list *<inst_list>*

Specifies the names of the blackboxed instances to be annotated with the given power numbers.

-unit_base *<fW | pW | nW | uW | mW | W>*

Specifies the unit of power numbers passed with `-internal_power` and `-leakage_power` options. The default power unit is `mW`.

Related Commands

report_power

Usage

Annotates precomputed internal power and leakage power on the blackboxed instances. User annotated power numbers are accounted for in power reports generated by PowerPro.

Examples

Example 1: Sets `.9mW` as the internal power consumption and `.1mW` as the leakage power consumption on the blackboxed instance `correlate`.

```
set_annotated_power -internal_power .9  -leakage_power .1
     -inst_list correlate
```

OR

```
set_annotated_power -internal_power .9  -leakage_power .1
    -inst_list {correlate}
```

Example 2: Sets `.9mW` as the internal power consumption and `.1mW` as the leakage power consumption on the blackboxed instances `correlate` and `rows`.

```
set_annotated_power -internal_power .9  -leakage_power .1
     -inst_list {correlate rows}
```

Example 3: Sets `.9mW` as the internal power consumption and `.1mW` as the leakage power consumption on the instance using the VHDL escape name `\correlate@rows\`.

```
set_annotated_power -internal_power .9  -leakage_power .1
     -inst_list {\correlate@rows\}
```

set_as_retiming_candidate

set_as_retiming_candidate — Specifies the modules or instances to be targeted for retiming during RTL synthesis. Note that this command can only be called after `prototype_design`.

Synopsis

`set_as_retiming_candidate -module <module_name> | -inst <instance_name>`

-module *<module_name>* | -inst *<instance_name>*

Use `-module` to mark all instantiations of the module *<module_name>* as candidates for retiming.

Use `-inst` to mark all instances named *<instance_name>* as candidates for retiming.

Related Commands

insert_observability_logic

Usage

The **set_as_retiming_candidate** command is used to specify modules for retiming during RTL synthesis.

set_banked_memory

set_banked_memory — Marks the specified instance or all instances of the specified module to be treated as a banked memory. The specified module must be a preserve_lib_module. Note that this command must be run after the `build_design` and before the `prototype_design` commands.

Synopsis

`set_banked_memory { -module <module> | -instance <instance> }`

-module *<module>*

The specified module whose all instances must be treated as banked memories.

-instance *<instance>*

The specified instance that must be treated as a banked memory.

Related Commands

insert_mem_observability_gating, insert_mem_stability_gating, preserve_lib_module

Usage

The **set_banked_memory** command marks the specified instance or all instances of the specified module as a banked memory. This command can be called only on the modules or instances that are already marked using the command 'preserve_lib_module'. This command must be used to generate gating conditions on the banked memories.

Examples

Example 1: Marks all instances of the `fast_mem` module as banked memories.

```
preserve_lib_module -memory -module fast_mem
build_design input.v
create_clock -name clk -period 20 clk
set_banked_memory -module fast_mem
prototype_design
insert_mem_observability_gating
insert_mem_stability_gating
```

Example 2: Marks only the `mem_unit.fast_mem_bank` instance of the `fast_mem` module as a banked memory.

```
preserve_lib_module -memory -module fast_mem
build_design input.v
create_clock -name clk -period 20 clk
set_banked_memory -inst mem_unit.fast_mem_bank
prototype_design
insert_mem_observability_gating
insert_mem_stability_gating
```

set_binding

set_binding — Specifies the technology cells to be used for mapping registers and CGICs in a hierarchical instance. Note that this command can only be called before `prototype_design`.

Synopsis

```
set_binding -library  <lib_name>  { -flop | -cgic }
    [ -pos <cell_name>] [ -neg <cell_name>]
    [ -latch_posedge_control <cell_name>]
    [ -latch_negedge_control <cell_name>]
    [ -none_posedge_control <cell_name>]
    [ -none_negedge_control <cell_name>]
    [ -inst <instance_list>]
```

-library *`<lib_name>`*

Specifies the name of the library in which the given technology cells are defined. This library must have been read using the `read_library` command.

-flop | -cgic

Specifies whether binding is to be applied to flops or CGICs.

Specify `-flop` to map the flops in the design using the technology cell specified. This option requires the `-pos` or `-neg` option to be specified.

Specify `-cgic` to map the CGICs in the design using the technology library read. This option requires that one of the following options be specified: `-latch_posedge_control`, `-latch_negedge_control`, `-none_posedge_control` and `-none_negedge_control`.

-pos *`<cell_name>`*

Specifies the name of the technology cell to be used for mapping positive edge triggered flops. Note that the technology cell specified must be a flop. This option requires the `-flop` option to be specified.

If the `-neg` option is not specified, the technology cell specified with `-pos` is used to map all negative edge triggered flops as well.

-neg *<cell_name>*

Specifies the name of the technology cell to be used for mapping negative edge triggered flops. Note that the technology cell specified must be a flop. This option requires the `-flop` option to be specified.

If the `-pos` option is not specified, the technology cell specified with `-neg` is used to map all positive edge triggered flops as well.

-latch_posedge_control *<cell_name>*

Specifies the name of the technology cell to be used for mapping CGICs of type `latch_posedge_control`. Note that the technology cell specified must be a CGIC with the `clock_gating_integrated_cell` attribute having a value `latch_posedge`. This option requires the `-cgic` option to be specified.

-latch_negedge_control *<cell_name>*

Specifies the name of the technology cell to be used for mapping CGICs of type `latch_negedge_control`. Note that the technology cell specified must be a CGIC with the `clock_gating_integrated_cell` attribute having a value `latch_negedge`. This option requires the `-cgic` option to be specified.

-none_posedge_control *<cell_name>*

Specifies the name of the technology cell to be used for mapping CGICs of type `none_posedge_control`. Note that the technology cell specified must be a CGIC with the `clock_gating_integrated_cell` attribute having a value `none_posedge`. This option requires the `-cgic` option to be specified.

-none_negedge_control *<cell_name>*

Specifies the name of the technology cell to be used for mapping CGICs of type `none_negedge_control`. Note that the technology cell specified must be a CGIC with the `clock_gating_integrated_cell` attribute having a value `none_negedge`. This option requires the `-cgic` option to be specified.

-inst *<instance_list>*

Specifies the list of hierarchical names of instances on which the constraint is to be applied. If this option is not specified, the constraint is applied to all design objects.

Related Commands

report_power, set_target_technology

Usage

The `set_binding` command lets you specify the technology cells that must be used for mapping flops and CGICs in a design. This command can be specified only after the `read_library` and before the `prototype_design commands`.

You can use the following syntax to specify binding for flops:

```
set_binding -library <lib_name>  -flop  [ -pos <cell_name>]
      [ -neg <cell_name>]  [ -inst <instance_name>]
```

where, either the `-pos` or the `-neg` option, or both can be specified. The <`cell_name`> argument can be a flop or a flop bank cell (also known as a multi-bit flop).

You can use the following syntax to specify binding for CGICs:

```
set_binding -library <lib_name>  -cgic
     [ -inst <instance_name> ]
     [ -latch_posedge_control <cell_name>]
     [ -latch_negedge_control <cell_name>]
     [ -none_posedge_control <cell_name>]
     [ -none_negedge_control <cell_name>]
```

At least one of the CGIC type options must be specified. For the CGIC types that have not been specified, PowerPro uses the minimum area cell for that type of CGIC.

Notes

- The `set_binding` command takes priority over the `set_target_technology` command.
- The technology libraries must be read using the `read_library` command before running the `set_binding` command.
- The `set_binding` command can be run only before the `prototype_design` command.

Examples

Example 1: Uses the technology flop `dff_1` to map all the positive edge triggered flops in the hierarchical instance `inst1`, and flop `ff_1` in the hierarchical instance `inst2`.

```
//file file1.lib
library(lib1)
{
....

cell(dff_1) {
  ff(IQ, IQN) {
    next_state : "D";
    clocked_on : "CK";
  }
....

//command
set_binding -library lib1 -pos dff_1 -inst "inst1 inst2.ff_1"
```

Example 2: Uses the flop bank `dffn_2` to map all the positive edge triggered flops in the hierarchical instance `hier1`.

```
//file file1.lib
library(lib1)
{
....

cell(dffn_2) {
  ff_bank(IQ, IQN, 2) {
    next_state : "D";
    clocked_on : "CK";
  }
....

//command
set_binding -library lib1 -pos dffn_2 -inst hier1
```

Example 3: Uses the technology flop `dff_1` to map all the positive edge triggered flops in the hierarchical instance `hier1`. Uses the flop bank `dffn_2` to map all the negative edge triggered flops in the hierarchical instance `hier1`.

```
//file file1.lib
library(lib1)
{
....

cell(dff_1) {
  ff(IQ, IQN) {
    next_state : "D";
    clocked_on : "CK";
  }
....

cell(dffn_2) {
  ff_bank(IQ, IQN, 2) {
    next_state : "D";
    clocked_on : "!CK";
  }
....

//command
set_binding -flop -library lib1 -pos dff_1 -neg dffn_2 -inst hier1
```

Example 4: Uses the technology cell `pos_cgic_1` to map all the positive edge triggered CGICs in the design. Uses the technology cell `neg_cgic_1` to map all the negative edge triggered CGICs in the design.

```
//file file1.lib
library(lib1)
{
...
cell(pos_cgic_1) {
  clock_gating_integrated_cell : "latch_posedge_postcontrol";
....
cell(neg_cgic_1) {
  clock_gating_integrated_cell : "latch_negedge_postcontrol";

//command
set_binding -cgic -library lib1 -latch_posedge_control pos_cgic_1
       -latch_negedge_control neg_cgic_1
```

set_bottom_up_flow

set_bottom_up_flow — Invokes the bottom-up flow and sets a submodule as the top hierarchy for the PowerPro run.

Synopsis

`set_bottom_up_flow` -top_module *<submodule_name>*

-top_module *<submodule_name>*

Specifies the name of the submodule to be set as the top module for the run.

Related Commands

write_rtl

Usage

Sets a submodule as the top hierarchy for the PowerPro run. This is particularly useful when different optimization objectives exist for each module or when the design runs into capacity issues. Broadly, the steps in a bottom up flow can be described as follows:

1. Prepare the Tcl script to point to the submodule.
2. Run the Tcl script.
3. Verify the design.
4. Run the `merge_bottomup_filelists` command to merge the modified files and filelist.

Note: Because the name of the top module or file does not change in the botto -up flow, it is important that the top module be defined in a separate file. Keeping the top module *in the same file as the other modules will lead to duplicate definitions at the time of the PowerPro run causing PowerPro to error out.*

Preparing the Tcl Script for the Bottom -up Flow

When performing step 1 above (preparing the Tcl script) make sure the following changes are made:

- The submodule is passed as an argument to the `build_design -top` option.
- The instance name passed with the `-instance_name` option of the `read_saif` and `read_fsdb` commands has been changed to point to the instantiation of the submodule.
- The clock definition has been changed to point to the clock ports of the submodule.
- All commands that apply instance-specific constraints like `create_mode_constraint`, `set_observable`, and `set_dont_care` have been changed to point to instances of the submodule.

Examples

Consider a design with the following hierarchical structure.

```
Top
    - BLK1
       - BLK1a
       - BLK1b
    - BLK2
       - BLK2a
       - BLK2b
    - BLK3
    - BLK4
```

Assume that different optimization objectives exist for each block in the hierarchy, the user would like to separately run PowerPro on BLK1, BLK2a and top (with BLK1 and BLK2a blackboxed). Further, assume that the user has a tcl script, `top.tcl` and a filelist `top.f` to run `top` in the normal flow. Given these assumptions, the user would need to perform the following steps:

- Prepare BLK1.tcl, BLK2a.tcl and top_others.tcl from top.tcl
- Run PowerPro on BLK1(using `BLK1.tcl` with workdir set to `ppro_BLK1`), BLK2a(using `BLK2a.tcl` with workdir set to `ppro_BLK2a`) and top (using `top_others.tcl` with workdir set to `ppro_top_others`).
- Run SLEC using `verify.tcl` on each of the `<workdir>/verify` directories.
- Merge the files from the various PowerPro runs using the script `merge_bottomup_filelists`. At the end of this step, a new RTL by the name of `rtl_mod.f` is created.
- Manually connect override ports from the subblocks to the top module.

Note: It is not mandatory to run PowerPro on the top level. The user can choose to run the bottom-up flow only on the relevant sub-blocks and then invoke `merge_bottomup_filelists` to merge the optimized RTL files.

---

set_boundary_opt

set_boundary_opt — Preserves the module interface so that new ports are not added to the module. The command can be called only after the `build_design` command and before the `prototype_design` command.

However, the override signals such as the CG/MG overrides, mode_overrides, and the reset signal can modify the module interface.

Note: Some QOR loss might be associated with the usage of this command.

Synopsis

`set_boundary_opt {-module <`*`module name`*`> | -instance <`*`hierarchical name`*`>}`

`-module <`*`module name`*`>`

Specifies the name of the module whose instances must be preserved.

`-instance <`*`hierarchical name`*`>`

Specifies the hierarchical name of the instance that must be preserved.

Related Commands

insert_mem_observability_gating, insert_mem_stability_gating, insert_observability_logic, insert_stability_logic

Usage

Preserves the module interface so that new ports are not added to the module.

Examples

Example 1: Preserves the instances of the module `xyz`.

```
set_boundary_opt -module xyz
```

Example 2: Preserves the hierarchical instance `abc.xyz`.

```
set_boundary_opt -instance abc.xyz
```

set_case_analysis

set_case_analysis — Assigns the constant value 0 or 1, to a list of ports or pins. Note that this command can only be called after `build_design` and before `prototype_design`.

Synopsis

```
set_case_analysis <value> <port_or_pin_list>
```

`<value>`

Specifies the constant value to be assigned. Legal values are `1`, `0`, `one` and `zero`.

`<port_or_pin_list>`

Specifies the list of ports or pins to which the constant value is to be assigned.

Related Commands

create_mode_constraint, set_logic_one, set_logic_zero

Usage

The **set_case_analysis** command is useful for specifying a given mode of the design without altering the design structure. The value assigned using the `set_case_analysis` command is propagated through the network as long as the controlling value for the traversed logic is at the constant value.

Notes:

- Such ports are excluded from the mode override logic created by PowerPro.
- This command should generally be used only for ports in the clock network.

Examples

Example 1: Assigns the constant value `0` to the port `func_mode`.

```
set_case_analysis 0 func_mode
```

set_cell_as_memory

Deprecated w.e.f. PowerPro 10.4.

set_cell_as_memory — Gathers memory clock gating information about ports on memories in the design. Memory clock gating information gathered by this command is used by the `insert_mem_clock_gating` command.

Synopsis

```
set_cell_as_memory { -module < module_name> }
    {-enable_polarity_clock <tuple(enable, enable_polarity, clock)>...}
    {-clock_and_polarity <tuple(clock, clock_polarity,
    CGIC_to_be_used)>...} [ -scan_signal <design_scan>]
```

-module *<module_name>*

Specifies the name of the module whose instances are to be marked for memory clock gating. Note that the module must be a technology memory cell or a power opt memory.

-enable_polarity_clock*<tuple(enable, enable_polarity, clock)>*...

Specifies a tuple corresponding to a memory enable pin, enable polarity followed by a clock pin of the specified module. Note that it is important to retain this order. Multiple enable -polarity -clock tuples can be specified by repeating this option.

-clock_and_polarity*<tuple(clock, clock_polarity, CGIC_to_be_used)>*...

Specifies a tuple of clock pin, clock polarity and the corresponding CGIC to be used for the module. Note that it is important to retain this order. Multiple clock -polarity -cgic tuples can be specified by repeating this option.

Note that *<CGIC_to_be_used>* is an optional value in this syntax. In case *<CGIC_to_be_used>* is not specified using this option, this value must be specified using the `set_target_cgic` command.

-scan_signal*<design_scan>*

Specifies the scan signal to be connected to the CGIC scan pin. In case *<design_scan>* is not specified using this option, this value can be specified using the `set_scan_enable` command.

Related Commands

insert_mem_clock_gating, report_clock_gated_memories, set_module_as_cgic, set_scan_enable, set_target_cgic

Usage

The **set_cell_as_memory** command gathers memory clock gating information about ports on memories in the design. Memory clock gating information gathered by this command is used by the `insert_mem_clock_gating` command.

Examples

Example 1: Marks instances of the module `p1024x32m` for clock gating. Here, the clock port `clk` will be gated during `insert_mem_clock_gating` with the respective `me` enable.

```
set_cell_as_memory  -module p1024x32m
     -enable_polarity_clock "me pos clk"  -clock_and_polarity "clk pos"
```

Note: If the module is not compiled or instantiated in the design, `set_cell_as_memory` exits with an error. Similarly, for the ports specified in the clock, enable tuples must be present in the module to successfully execute the command.

set_cg_override

set_cg_override — Sets the override port or signal in the design. Note that this command can be run only before any sequential optimization command is run.

Note: In most cases, the command is run after the `prototype_design` command.

However, in some cases, it can be run before the `prototype_design` command as well. For example, if the output of a design object, such as a flop or a hierarchical instance, does not drive any other logic and must be used as an override, then, the command can be run before the `prototype_design` command. The design objects that are used as overrides are preserved during the `prototype_design` phase and are not removed by logic optimizations.

Synopsis

```
set_cg_override { -port <port_name> | -signal <signal_name> } [ -auto ]
    [ -target_clock <clock_name> ] [ -target_polarity <pos | neg> ]
    [ -ff_list <flop_list> | -inst_list <instance_list> ]
```

-port *<port_name>* | -signal *<signal_name>*

Specifies the name of the port or signal to be used as the override port or signal. If the specified port or signal is not present in the design, PowerPro exits with an error.

To set an existing port as the override port, the port must be a top hierarchy input port or an output port other than the top hierarchy port. Likewise, to set an existing signal as the override signal, it should be driven by a top hierarchy input port or an output port other than the top hierarchy port.

To automatically create a port and set it as the override port, use the `-auto` option.

-auto

Specifies that the port specified by `-port` be automatically created in the top module in case it does not exist in the design.

-override_inst *<override_inst_name>*

The `-override_inst` option has been deprecated w.e.f PowerPro 7.1.

-target_clock *<clock_name>*

Specifies that the port or signal be used as an override port/signal only for those flip flops, which are driven by the clock *<clock_name>*.

-target_polarity *<polarity>*

Specifies that the port or signal be used as an override port or signal only for those flip flops whose clock network has the specified polarity. Legal polarity values are `pos` and `neg`.

Note that this option will be honored only if the `-target_clock` option has also been specified.

-ff_list *<flop_list>* | -inst_list *<instance_list>*

Use `-ff_list` to specify the list of flip flops to override. Note that the names provided must be hierarchical.

Use `-inst_list` to specify the list of instances whose flops this command will override.

If neither of these options are specified, `set_cg_override` overrides all flops in the design with the specified port/signal.

Related Commands

set_mem_override

Usage

The **set_cg_override** command sets the override port or signal for individual flops in the design. When specified, PowerPro creates a new instance and uses the output port of this instance as the CG override port. Note that the clock gating logic can be disabled by setting the override port or signal to `1` in the RTL.

Examples

Example 1: Sets the port `override` as the override port for all flops in the design. If the override port `override` does not exist, PowerPro exits with an error.

```
set_cg_override -port override
```

Example 2: Sets the signal `override` as the override signal for all flops in the design. If the override signal `override` does not exist, PowerPro exits with an error.

```
set_cg_override -signal override
```

Example 3: Sets the port `override` as the override port for all flops in the design. In case the override port `override` does not exist, PowerPro automatically creates a port by this name in the top module.

```
set_cg_override -port override -auto
```

Example 4: Sets the port `override` as the override port only for the flops `a.f1` and `b.f2`. In case the override port `override` does not exist, PowerPro automatically creates a port by this name in the top module.

```
set_cg_override -port override -auto -ff_list "a.f1 b.f2"
```

Example 5: Sets the port `override` as the override port only for the flops inside the instance `a.b`. In case the override port `override` does not exist, PowerPro automatically creates a port by this name in the top module.

```
set_cg_override -port override -auto -inst_list "a.b"
```

Example 6: Sets the port `override` as the override port for all flops that have a positive network polarity and are driven by the clock domain `CLK`. In case the override port `override` does not exist, PowerPro automatically creates a port by this name in the top module.

```
create_clock -name CLK -period 123.45 clk
set_cg_override -port override -auto -target_clock CLK
   -target_polarity pos
```

set_cgic_cell

set_cgic_cell — Replaces user clock gating modules with PowerPro clock gating modules. Note that this command can only be called between `build_design` and `prototype_design`.

Synopsis

```
set_cgic_cell  -user_cell <user_clock_gating_module>
     -calypto_cell <powerpro_clock_gating_module>
```

-user_cell *<user_clock_gating_module>*

Specifies the name of the user clock gating module.

-calypto_cell *<powerpro_clock_gating_module>*

Specifies the name of the PowerPro clock gating module. Note that the interface of this module must be exactly the same as that of the user clock gating module.

Related Commands

build_design, write_cgic_module

Usage

The **set_cgic_cell** command replaces user clock gating modules with PowerPro clock gating modules. This command can only be called after `build_design` and before `prototype_design`. In case `read_saif` is also being called in the flow, `set_cgic_cell` must be called before `read_saif`.

Examples

Example 1: Replaces the user clock gating module `user_cgic_cell_0` with the PowerPro clock gating module `calypto_cgic_cell_0`. Likewise, `user_cgic_cell_1` is replaced with `calypto_cgic_cell_1`. Note that the `set_cgic_cell` command is called before the `read_saif` command.

```
build_design test.v
build_design  -cons cgic_cell.v

set_cgic_cell  -user_cell user_cgic_cell_0  -calypto_cell
calypto_cgic_cell_0
set_cgic_cell  -user_cell user_cgic_cell_1  -calypto_cell
calypto_cgic_cell_1

read_saif test.saif
prototype_design
```

Example 2: This is an example of incorrect usage. Here, the `set_cgic_cell` command is called between `read_saif` and `prototype_design`. Calling `set_cgic_cell` in the wrong sequence will lead to an error.

```
read_saif test.saif
set_cgic_cell  -user_cell user_cgic_cell_0  \
     -calypto_cell calypto_cgic_cell_0
set_cgic_cell -user_cell user_cgic_cell_1 \
     -calypto_cell calypto_cgic_cell_1
prototype_design
```

set_clock_domain

set_clock_domain — Specifies the clock association and polarity for the given port or signal. Note that the specified port or signal must be a primary input or a black box output.

Synopsis

```
set_clock_domain  -clock <clock_name>
    [ -port <port_name> ... |  -signal <signal_name> ... ]
    [ -polarity <pos|neg> ... ]
```

-clock *<clock_name>*

Specifies the name of the clock.

-port *<port_name>*...

Specifies the name of the primary input or black box output port.

-signal *<signal_name>*...

Specifies the name of the primary input or black box output signal.

-polarity *<pos|neg>*...

Specifies the polarity of the specified clock. If not specified, the polarity is assumed to be positive.

Related Commands

create_clock, set_powerpro_reset

Usage

The **set_clock_domain** command specifies the clock association and polarity of the given port or signal. Note that the specified port or signal must be a primary input or black box output.

Examples

Example 1: Associates the clock domain `clk` with the primary input port `in1`

```
set_clock_domain  -clock clk  -port in1
```

Example 2: Associates the clock domain `clk` with the primary input signal `in1`

```
set_clock_domain  -clock clk  -signal in1
```

Example 3: Associates the negative edge of the clock domain `clk` with the signal `in1`

```
set_clock_domain  -clock clk  -signal in1  -polarity neg
```

set_current_hierarchy

set_current_hierarchy — Sets the specified hierarchal instance as the current working hierarchy. This means that the data must be reported only for the specified hierarchy and the hierarchies below it. Note that this command can be run only after the `build_design` command and before the `prototype_design` command.

Synopsis

`set_current_hierarchy <`*`instance_name`*`>`

<*`instance_name`*>

Specifies the hierarchical name of the instance that must be set as the current working hierarchy.

Related Commands

report_power, report_enable_logic, report_memory_gating

Usage

The `set_current_hierarchy` command can be used when, for example, the setup for the complete design is available, however, we have to run PowerPro only on a specific hierarchy. The command sets the specified hierarchal instance as the current working hierarchy. Once the current working hierarchy is set, the reporting and optimization commands are run on this hierarchy.

Example

Example 1: Consider a design, where three blocks `inst_blockA`, `inst_blockB`, and `inst_blockC` are under the top module `soc_top`. In this example, the following command can be used to set the hierarchal instance `inst_blockA` as the current working hierarchy.

```
set_current_hierarchy inst_blockA
```

set_default_threshold_voltage_group

set_default_threshold_voltage_group — Sets the `default_threshold_voltage_group` attribute on the specified libraries. Note that this command can only be called between `read_library` and `prototype_design`.

Synopsis

```
set_default_threshold_voltage_group -name <group_name>
    -library <library_name> ...
```

-name *<group_name>*

Specifies the name of the threshold voltage group to set as the `default_threshold_voltage_group` attribute for the given library. All cells that do not have the `threshold_voltage_group` attribute will be assigned to this group.

-library *<library_name>* ...

Specifies the name of the library/libraries on which the `default_threshold_voltage_group` attribute is to be set.

Related Commands

set_multi_vth_constraint, set_threshold_voltage_group

Usage

The `set_default_threshold_voltage_group` command sets the `default_threshold_voltage_group` attribute on the specified libraries.

Examples

Example 1: Specifies that all cells in the library `lib1` which do not have the `threshold_voltage_group` attribute specified be assigned to the group `DEFAULT`.

```
set_default_threshold_voltage_group -name "DEFAULT" -library lib1
```

Note: The name DEFAULT is not a legal value and holds no significance. This can be any string (See Example 2).

Example 2: Sets `abc` as the `default_threshold_voltage_group` attribute on the the libraries `lib1` and `lib2`.

```
set_default_threshold_voltage_group -name "abc" -library "lib1"
     -library "lib2"
```

set_design_scope

set_design_scope — Sets the design scope for the top module. This is useful when the top module requires additional information from its containing hierarchy. Note that this command can be called only before the `link_design` or the `build_design` command.

Synopsis

```
set_design_scope <design_scope> [-target_instance <inst>]
```

<design_scope>

Name of the containing hierarchy that has the additional context information required by the top module.

-target_instance <*inst*>

The hierarchical instance name upto the instance of the design top in the design scope.

Related Commands

build_design

Usage

The usage of the `set_design_scope` command is explained below:

- The `set_design_scope` command must be issued before linking the design.
- The `set_design_scope` command cannot be issued if the `-defparam` option was used while linking the design.
- If multiple instances of the top module are present in the design scope hierarchy, the `-target_instance` option can be used to specify the name of the instance using which the additional context information can be obtained.

Examples

Example 1: Sets the design scope for the top module. Consider the following example.

```
module bot1 (parameter X =2);
.....
.....
endmodule

module test;
  bot1 #(.X(6)) inst1( i1, clk);
  bot2 #(.Y(7)) inst2( clk );
endmodule
```

In this example, the parameter `X` in the module `bot1` has a default value `2`. Assume that the top module is `test`. In this scenario, if the module `bot1` was to be set as the top module using the `build_design -top` command, `i1` would be passed with its default parameter value `2` and not the value `6` as defined within the module test.

The `set_design_scope` command allows the user to pass the parameter value `6` for the module `bot1`, as follows:

```
set_design_scope test
build_design -top bot1 design.sv
```

Running these commands will set `bot1` as the top module with interface port values from the module `test`.

Example 2: In the following code snippet, the module `bot1` has multiple instances `inst1` and `inst2`.

```
module bot1 (parameter X =2);
.....
.....
endmodule

module test;
  bot1 #(.X(6)) inst1( i1, clk);
  bot1 #(.X(7)) inst2( i1, clk);
endmodule
```

In this case, the additional context information for the instance `inst1` can be obtained using the `–target_instance` option as shown below:

```
set_design_scope test –target_instance inst1
build_design -top bot1 design.sv
```

set_dont_care

set_dont_care — Sets the dont-care expression on the specified list of ports. Note that this command can only be called after `prototype_design`.

Synopsis

```
set_dont_care {  -when <port> |  -when_not <port> }
    [ -transactor <transactor_name>  -port_map <port_map_string>]
    <list_of_ports>...
```

-when *<port>* | -when_not *<port>*

Specifies the name of the dont-care expression output port and its polarity. When this option is specified with the `-transactor` option, *<port>* must refer to the output port of the transactor module. When specified standalone, it must refer to a signal in the RTL.

Note: *<port>* must be a primary output port or a blackbox input port, when the `–transactor` *<transactor_name>* option is specified.

-transactor *<transactor_name>* -port_map *<port_map_string>*

<transactor_name> refers to the name of the module containing the dont-care expression while *<port_map_string>* refers to the expression to map input ports of the transactor module to the signals in the RTL.

<port_map_string> must be specified in the following format:

```
{ .<transactor_input_port_1> (<design_signal_1>), ...,
.<transactor_input_port_n> <design_signal_n>),
[.<transactor_output_port_for_when_or_when_not> ()] }
```

Notes:

- The transactor must have been read in earlier using the `build_design -cons` command.
- The transactor must not have been instantiated anywhere in the design.
- The name of the transactor must not clash with an existing hierarchy.
- Signals specified for mapping must be primary outputs or black box inputs.
- The output port is left unmapped.

<list of ports>...

A whitespace separated target list of RTL ports for which the dont-care setting is to be applied.

Related Commands

insert_mem_observability_gating, insert_mem_stability_gating, insert_mem_sleep_gating, insert_observability_logic, insert_stability_logic

Usage

The **set_dont_care** command sets the dont-care expression on the specified list of target ports.

Notes:

- The when/when_not port, the target port and the design ports specified in the *<port_map_string>* must be a primary output port or a black box input port.
- The ports specified with when/when_not and target cannot be same even across different calls to this command.
- The design ports specified in the *<port_map_string>* and the target cannot be the same even across different calls to this command.

Examples

Example 1: Sets `s1` as the output dont-care expression for the signals `e1`, `e2` and `e3`, where `s1`, `e1`, `e2` and `e3` are primary output or blackbox input ports in the design.

```
set_dont_care  -when s1 e1 e2 e3
```

Example 2: Sets the port `!M1.FF.q` as the dont-care expression for signals attached to the hierarchical port `M1.p1`.

```
set_dont_care  -when_not M1.FF.q  M1.p1
```

Example 3: Sets the port `sel` as the dont-care expression for signals attached to the sliced port `o3`.

```
set_dont_care  -when sel[0] o3[4:0]
```

Example 4: Sets the expression in the fanin cone of the output port `out` of the transactor module `trans` as the dont-care expression for the port `p1`. In this example, `in1` and `in2` are the input ports of the transactor `trans` while `y` and `z` are the corresponding design ports.

```
set_dont_care  -when out  -transactor trans  -port_map {.in1(y),
.in2(z)} p1
```

Example 5: Sets the dont-care constraint on the port `p1` using the dont-care expression specified by the transactor module `invert`. Assumptions are as follows:

- The transactor `trn39.v` contains the module `invert`.

  ```
  module invert (output out1, input in1);
    assign out1 = ~in1;
  endmodule
  ```

- The effective dont-care expression on the port `p1`, in this example, is `= out1 = ~in1 = ~s1 .`

The commands specified would be as follows:

```
build_design  -cons trn39.v
prototype_design
set_dont_care  -when out1  -transactor invert  -port_map {.in1(s1)} p1
```

set_dont_optimize

set_dont_optimize — Marks flops for exclusion from sequential optimization. Note that this command can only be called after `prototype_design`.

Synopsis

```
set_dont_optimize [ -observability ] [ -stability ] [ -stability_c ]
    [ -stability_s ] [ -all ] [ -reset]  -inst <instance_list>
```

-observability

Specify `-observability` to exclude the instances specified by `-inst` from observability-based sequential optimization. Note that this option can only be called before calling the `insert_observability_logic` command.

-stability

Specify `-stability` to exclude the instances specified by `-inst` from stability-based sequential optimization (constant and symbolic). Note that this option can only be called before calling the `insert_stability_logic` command.

-stability_c

Specify `-stability_c` to exclude the instances specified by `-inst` from constant stability-based sequential optimization. Note that this option can only be called before calling the `insert_stability_logic  -type C` command.

-stability_s

Specify `-stability_s` to exclude the instances specified by `-inst` from symbolic stability-based sequential optimization. Note that this option can only be called before calling the `insert_stability_logic  -type S` command.

-reset

Specify `-reset` to remove dont-optimize constraints already applied to the instances specified by `-inst`.

-all

Specify `-all` to exclude the instances specified by `-inst` from both observability and stability-based sequential optimization. This is the default.

-inst *<instance_list>* ...

A whitespace separated list of instances that are to be excluded from the specified optimization. In case of a hierarchical instance, all flops within the instance are excluded from optimization.

Related Commands

insert_observability_logic, insert_stability_logic, set_mem_optimize, set_mem_dont_optimize

Usage

The **set_dont_optimize** command excludes flops from optimization. For hierarchical instances, all flops within the hierarchy are excluded.

Examples

Example 1: Excludes the instances `M1.ff1` and `M1.ff2` from observability-based sequential optimization.

```
prototype_design
...
set_dont_optimize  -inst "M1.ff1 M1.ff2"  -observability
...
insert_observability_logic
```

Example 2: Removes the dont-optimize constraint from the instances `M1.ff1` and `M1.ff2`.

```
set_dont_optimize  -inst "M1.ff1 M1.ff2"  -reset
```

set_dont_use

set_dont_use — Sets the specified dont-use constraint on the list of signals/hierarchical instances OR the `dont_use` attribute on the technology library cells provided. Note that, with the `-signal` or `-inst` options, this command can only be called after `prototype_design`. If the `-lib_cell` option is specified, this command can only be called before `prototype_design`.

Synopsis

```
set_dont_use {[-inst <instance_list> | -signal <signal_list>] |
    -lib_cell <lib_cell_list> } [-seq_fanout]
    [ -observability | -stability | -all ]
```

-inst *<instance_list>* | -signal *<signal_list>*

A whitespace separated list of hierarchical instances or signals on which the dont-use constraint is to be applied. In case a hierarchical instance is specified, all signals inside that hierarchy are marked. When either of these options are specified, the `-lib_cell` option cannot be specified.

-lib_cell *<lib_cell_list>*

A whitespace separated list of technology library cells in the *<lib_name/cell_name>* format for which the `dont_use` attribute is to be applied. Technology library cells specified in this list will not be used for mapping. When this option is specified, all other options are ignored.

-seq_fanout

Sets the specified dont-use constraint in the fanout cone of the given signals until the first sequential depth. If there is no sequential element in the fanout, the dont-use constraint will be set on all signals until a primary output or blackbox input is reached. Note that this option can only be used with the `-signal` option.

-observability | -stability | -all

Specifies the type of sequential optimization for which the dont-use constraint is to be applied. In case the type of optimization is not specified, the hierarchical instance or signal specified is excluded from the clock gating hierarchy created during observability and stability based sequential optimization.

Specify `-observability` to exclude the specified signal or hierarchical instance from the clock gating hierarchy created during observability-based sequential optimization.

Specify `-stability` to exclude the specified signal or hierarchical instance from the clock gating hierarchy created during stability-based sequential optimization.

Specify `-all` to exclude the specified signal or hierarchical instance from both observability-based and stability-based sequential optimization.

Related Commands

dont_use_high_fanout_signals, dont_use_high_sequential_depth_signals, find_lib_cell, insert_observability_logic, insert_stability_logic, insert_mem_observability_gating, insert_mem_sleep_gating, insert_mem_stability_gating, parse_and_mark_timing_critical_paths

Usage

The `set_dont_use` command can be used in two ways:

- To apply the dont-use constraint on design objects
- To apply the `dont_use` attribute on technology library cells

The `set_dont_use -signal/-inst` command applies the dont-use constraint on signals/instances. Signals/instances marked with the dont-use constraint are excluded from the clock gating hierarchy created during sequential optimization. The `dont-use` constraint is most useful when a signal is to be explicitly excluded from a clock gating or memory gating expression.

Consider the following example:

```
module top(input [7:0] in1, in2, input en1, en2, clk, rst,
           output [7:0] out1, out2);

reg [7:0] in_p1, in_p2;
reg       en_p1, en_p2;
wire      flop_in;

assign flop_in = en1 & en2;
assign out1 = en_p1 ? in_p1 : 8'b0;
assign out2 = en_p2 ? in_p2 : 8'b0;

always @(posedge clk)
begin
    in_p1 <= in1;
    en_p1 <= en1;

    in_p2 <= in2;
    en_p2 <= flop_in;
end

endmodule
```

In this example, when the `set_dont_use` command is applied on the signal `flop_in`, the flop `en_p2` is automatically excluded from clock gating. This is because the signal `flop_in` does not participate in the creation of the clock gating hierarchy during sequential optimization.

```
set_dont_use -signal flop_in
```

Note: The `set_dont_use` command cannot be applied to PowerPro generated signals or nets that do not exist as signals or nets in user's design RTL. Using the `set_dont_use` command with such signals will lead to an error.

When the `set_dont_use -lib_cell` command is specified, the `dont_use` attribute is applied to the technology library cells. Such technology cells are excluded from mapping. For example, the following command applies the `dont_use` attribute on the cells `AND1` and `OR1` belonging to the library `lib1`

```
set_dont_use -lib_cell  "lib1/AND1 lib1/OR1"
```

Note that if the `-inst`, `-signal` or `-lib_cell` option is not specified but a list is specified with the `set_dont_use` command, the list is assumed to be a list of technology library cells on which the `dont_use` attribute is to be applied. The following command is equivalent to the above mentioned example.

```
set_dont_use "lib1/AND1 lib1/OR1"
```

Examples

Example 1: Excludes the signal `shift` from observability based sequential optimization.

```
set_dont_use -signal shift -observability
```

Example 2: Excludes the signal `shift` from both observability-based and stability-based sequential optimization.

```
set_dont_use -signal shift
```

Example 3: Excludes all signals in the instance `decode_row` from sequential optimization.

```
set_dont_use -inst decode_row
```

Example 4: Starting with the signal `shift`, excludes all signals until the first sequential depth is reached.

```
set_dont_use -signal shift -seq_fanout
```

Example 5: Applies the `dont_use` attribute to the technology cell `ff1` from the library `lib3`. These cells will be excluded from mapping.

```
set_dont_use -lib_cell lib3/ff1
```

set_existing_synchronizer

set_existing_synchronizer — Specifies a list of modules that are instantiated as synchronizers in the design or a list of signals that are already synchronized. Note that this command can only be called after `build_design` and before calling `insert_observability_logic` or `insert_mem_observability`.

Synopsis

```
set_existing_synchronizer
    -module <list_of_modules>  -signal <list_of_signals>
```

-module *<list_of_modules>*

Specifies the names of the synchronizer modules in the design.

-signal *<list_of_signals>*

Specifies a list of synchronized signals.

Related Commands

mark_synchronizer

Usage

The `set_existing_synchronizer` command specifies the list of modules that are instantiated as synchronizers in the design or a list of signals that have already been synchronized. While creating the Asynchronous Override Logic (AOL), the output of the synchronizer modules or the already synchronized signals will not be synchronized but will be used directly as inputs to the AOL.

Examples

Example 1: In this example, `sync_reset1` and `sync_reset2` are synchronizer modules while `sig1` and `sig2` are already synchronized signals. If the signals `sig1` and `sig2` or the outputs of instances of the synchronizer modules `sync_reset1` and `sync_reset2` are considered for the AOL logic, a synchronizer will not be created on these signals.

```
set_existing_synchronizer  -module {sync_reset1 sync_reset2}  \
    -signal {sig1 sig2}
```

set_frequency

set_frequency — Annotates a port or signal with the specified frequency information. This command can only be called after the `build_design` and before the `prototype_design` command.

Synopsis

```
set_frequency {-ports <port_names_list> | -
    signals <signal_names_list> } { -frequency <value> }
```

-ports <*port_names_list*> | -signals <*signal_names_list*>

Specifies the type of object that must be annotated with the specified frequency information. The object must be specified as the full hierarchical name.

Specify `-ports` to annotate a port or a list of ports with the specified frequency information.

Specify `-signals` to annotate a signal or a list of signals with the specified frequency information.

-frequency <*value*>

Specifies the frequency in Megahertz, where the <*value*> is a floating point number greater than `0`.

Related Commands

set_sa, get_sa, pget design.insts.frequency

Usage

The `set_frequency` command can be used to set the frequency for the specified signal(s) or port(s).

Examples

Example 1: The following command sets the frequency of `123.45 MHz` on the signal `clk`.

```
set_frequency -signal clk -frequency 123.45
```

Example 2: The following command sets the frequency of `500 MHz` on signals `n1`, `n2`, and `n3`.

```
set_frequency -signals {n1 n2 n3} -frequency 500
```

Example 3: The below command sets the frequency of `500 MHz` on the port `flop.clk`.

```
set_frequency -port flop.clk -frequency 500
```

set_global

set_global — Sets the value for a specific global.

Synopsis

`set_global` *`<global_name>`* *`<value>`*

`<global_name>`

Specifies the name of the global whose value is to be set.

`<value>`

Specifies the value to be set.

Related Commands

get_global

Usage

Globals are used to alter the default behavior of PowerPro. Use the `set_global` command to set the value of a specific global.

Examples

Example 1: Sets the value of the global `opt_cg_min_size` to `12`.

```
set_global opt_cg_min_size 12
```

---

set_gray_box

set_gray_box — Applies the dont-optimize and dont-use constraints respectively on all flops and signals in the specified instance.

Synopsis

```
set_gray_box {  -module <module_name> |  -inst <instance_name> }
```

-module *`<module_name>`*

> Specifies the name of the module whose instances must have the dont-optimize and dont-use constraints applied on all signals and flops respectively.
>
> Note that this option can be specified only before the `build_design` command.

-inst *`<instance_name>`*

> Specifies the name of the instance for which the dont-optimize and dont-use constraints are applied on all signals and flops respectively.
>
> Note that this option can be specified only after the `build_design` command.

Related Commands

insert_observability_logic, insert_stability_logic, insert_mem_observability_gating, insert_mem_sleep_gating, insert_mem_stability_gating

Usage

The **set_gray_box** command does not optimize flops within the module or instance specified. Additionally, signals within the module/instance are not used to create PowerPro enables.

Dont-care conditions in the fanout of grayboxed modules are propagated to the fanin and help clock-gate the registers in the fanin. Stability conditions, on the other hand, are forward propagated across grayboxed instances.

Note that the `set_gray_box` command supports the use of wildcards and regular expressions.

Examples

Example 1: Grayboxes all hierarchical instances named DW*.

```
set graybox_inst [find  -hier  -inst DW*]
set_gray_box  -inst $graybox_inst
```

Example 2: Grayboxes all instances of the module `synchronizer_module`. This command ensures that signals within the synchronizer module, which may be metastable at times, are not used for generating PowerPro enables. It also ensures that PowerPro does not add enables to flops contained within the synchronizers.

```
set_gray_box  -module synchronizer_module
```

set_ignore_signal

set_ignore_signal — Excludes the specified signals/ports from sequential analysis. Note that this command can only be called after `prototype_design`.

Synopsis

```
set_ignore_signal {  -port <port>... |  -signal <signal>... |
      -instance <instance>... } [ -observability | -stability | -all]
```

-port *<port>* ... | -signal *<signal>* ... | -instance *<instance>* ...

Instructs PowerPro to exclude the specified port, signal or instance from sequential analysis.

Specify `-port` to exclude all signals associated with the port from sequential analysis. To exclude multiple ports, the `-port` option must be specified for each port.

Specify `-signal` to exclude the signal from sequential analysis. To exclude multiple signals, the `-signal` option must be specified for each signal.

Specify `-instance` to exclude all signals within the instance and its containing hierarchies from sequential analysis. To exclude multiple instances, the `-signal` option must be specified for each signal.

-observability | -stability | -all

Specifies the type of sequential analysis from which the signal/port/instance is to be excluded. In case the type of analysis is not specified, the hierarchical instance or signal specified is excluded from both observability-based and stability-based sequential analysis.

Specify `-observability` to exclude the specified signal or hierarchical instance from sequential analysis.

Specify `-stability` to exclude the specified signal or hierarchical instance from the clock gating hierarchy created during stability-based sequential analysis.

Specify `-all` to exclude the specified signal or hierarchical instance from both observability-based and stability-based sequential analysis. This is the default.

Related Commands

insert_observability_logic, insert_stability_logic, insert_mem_observability_gating, insert_mem_sleep_gating, insert_mem_stability_gating, set_dont_use

Usage

The `set_ignore_signal` command excludes the specified signals/ports from sequential analysis. Excluding a signal from analysis speeds up optimization, reducing capacity issues. Signals typically excluded include resets and ECC logic.

Consider the following example.

```
module top(input [7:0] in1, in2, input en1, en2, clk, rst,
           output [7:0] out1, out2);

reg [7:0] in_p1, in_p2;
reg       en_p1, en_p2;
wire      flop_in;

assign flop_in = en1 & en2;
assign out1 = en_p1 ? in_p1 : 8'b0;
assign out2 = en_p2 ? in_p2 : 8'b0;

always @(posedge clk)
begin
    in_p1 <= in1;
    en_p1 <= en1;

    in_p2 <= in2;
    en_p2 <= flop_in;
end

endmodule
```

In this example, applying the `set_ignore_signal` command to the signal `en_p2` will discard the clock gating opportunity on the flop `in_p2` as the signal `en_p2` will no longer remain as the originating point for sequential analysis.

```
powerpro> set_ignore_signal  -signal en_p2
```

Note: In the same example, if the dont-use constraint is applied on the signal `en_p2` (instead of the ignore -signal constraint), the flop `in_p2` will not be discarded and will continue to be part of sequential optimization. Refer to `set_dont_use` for more information.

Examples

Example 1: Excludes the signal `M1.s1` from sequential analysis.

```
set_ignore_signal  -signal M1.s1
```

Example 2: Excludes all signals driven by the port `M1.p1` from sequential analysis.

```
set_ignore_signal  -port M1.p1
```

set_input_delay

set_input_delay — Sets the input delay on specified input ports relative to the active edge of the specified clock signal.

Synopsis

```
set_input_delay  -clock <named_clock> [ -rise ] [ -fall ]
    <delay_value> <port>...
```

-clock *<named_clock>*

Specifies the name of the clock signal attached to the specified delay.

-rise

Specifies that the input delay is to be applied relative to the rising transition of the named clock signal.

-fall

Specifies that the input delay is to be applied relative to the falling transition of the named clock signal.

<delay_value>

Specifies the time, in nanoseconds, after which the signal arrives at the input port after the appropriate edge of the named clock signal. *Note that the delay value must be specified before the port names are specified.*

<port>...

Specifies the list of primary input ports on which the constraint is to be applied. Note that these must be primary inputs. No other hierarchical ports are allowed.

Related Commands

create_clock, set_clock_domain

Usage

The **set_input_delay** command specifies the arrival time of the input port(s) relative to the edge of the named clock signal. If the command is not specified on a given primary input port, the port is assumed to be unconstrained relative to the named clock signals in the design. If the command is specified with no ports, then the input delay is applied to all primary input ports in the design (except previously -designated clock roots set through the **create_clock** command).

If the type of transition is not specified, the input delay is assumed to be relative to both the rising and falling transition of the named clock signal.

Examples

Example 1: Assigns an input delay of 0.2 ns on input `in1` with respect to the rising transition of the clock signal `foo_clock`.

```
set_input_delay  -rise  -clock foo_clock 0.2 {in1}
```

Example 2: Assigns an input delay of 0.4 ns on all the primary inputs of the design relative to the rising and falling transitions of the clock signal `foo_clock`. Note that primary inputs designated as clock roots are not affected.

```
set_input_delay  -clock foo_clock 0.4 [all_inputs]
```

set_library_path

set_library_path — Replaces the path to libraries in a previously written database file. This command can only be called before `read_db` or `read_eco_db`.

Synopsis

```
set_library_path <tcl_file>
```

`<tcl_file>`

A Tcl file containing `read_library` commands pointing to the new library paths.

Usage

The **set_library_path** command works in conjunction with the `read_db` and `read_eco_db` commands to replace the library paths in an existing database file. This is useful when the paths to the libraries have changed since the last `write_db` session. This command expects a Tcl file as its argument - where, the Tcl file contains `read_library` commands pointing to the new library paths. A sample Tcl file is shown below.

```
read_library  new_path1/old_lib1.tcl
read_library  new_path2/old_lib2.tcl
read_library  new_path3/old_lib3.tcl
```

Examples

Example 1: Replaces the paths in the database file, `my_db.db` with the paths specified in `paths.tcl`.

```
set_library_path paths.tcl
read_db my_db.db
```

set_load

set_load — Sets a load value for the specified port or signal. Note that this command can only be called after `build_design` and before `prototype_design`.

Synopsis

`set_load` *`<value>`* *`<objects>`*

`<value>`

Specifies the load value to be applied on the given port(s) or signal(s).

Note: The load value must be defined in terms of the same unit used by the technology library used during optimization. For example, if the technology library specifies load values in `picofarads(pF)`, *`<value>`* must also be expressed in `picofarads(pF)`.

`<objects>`

Specifies the list of ports and signals in the current design whose loads are to be set.

Note: The value specified by *`<objects>`* must be a design output port, a signal connected to a design output port, a black box input port, or a signal connected to a black box input port.

Related Commands

set_powerpro_mode

Usage

The `set_load` command specifies the load value to be applied on the given port(s) or signal(s).

Examples

For all these examples, `out_port` is assumed to be a 7-bit design output port, with a precomputed (by tool) value of 0.5555 for each bit.

Example 1: `set_load` overrides the value of `0.5555` with `1.123434` for all 7 bits of the design port `out_port`

```
set_load  1.123434 out_port
Result: 1.123434 1.123434 1.123434 1.123434 1.123434 1.123434 1.123434
```

Example 2: `set_load` overrides the value `0.5555` stored in the 0th bit of `out_port` with `1.123434`

```
set_load 1.123434 out_port[0]
Result: 1.123434 0.5555 0.5555 0.5555 0.5555 0.5555 0.5555
```

Example 3: `set_load` overrides the values in the 0th, 1st and 2nd bits of `out_port` with `1.123434`

```
set_load 1.123434 out_port[0:2]
Result: 1.123434 1.123434 1.123434 0.5555 0.5555 0.5555 0.5555
```

Likewise, issuing the following command after the above mentioned command:

```
set_load 0.121212 out_port[3:4]
```

will override the values in the 3rd and 4th bit of `out_port` with 0.121212 and generate the following results:

```
1.123434 1.123434 1.123434 0.121212 0.121212 0.5555 0.5555
```

A message is also displayed indicating that user-asserted load values will be overwritten.

set_logic_one

set_logic_one — Drives one or more input ports in the design by the logic one. Note that this command can only be called before `prototype_design`.

Synopsis

`set_logic_one <port_or_pin_list>`

`<port_or_pin_list>`

Specifies the input port or pin list to which the value `1` is to be assigned.

Related Commands

set_case_analysis, set_logic_zero

Usage

The `set_logic_one` command drives one or more input ports in the design by the logic one. Note that this will lead to a reduction in the design as logic that has already been traversed will be eliminated.

Examples

Example 1: Drives the input ports `a`, `b`, and `c` by the logic one.

```
set_logic_one {a b c}
```

set_logic_zero

set_logic_zero — Drives one or more input ports in the design by the logic zero. Note that this command can only be called before `prototype_design`.

Synopsis

`set_logic_zero` *`<port_or_pin_list>`*

`<port_or_pin_list>`

Specifies the input port or pin list to which the value `0` is to be assigned.

Related Commands

set_case_analysis, set_logic_one

Usage

The `set_logic_zero` command drives one or more input ports in the design by the logic zero. Note that this will lead to a reduction in the design as logic that has already been traversed will be eliminated.

Examples

Example 1: Drives the input ports `a`, `b`, and `c` by the logic zero.

```
set_logic_zero {a b c}
```

set_mem_dont_optimize

set_mem_dont_optimize — Marks the memories for exclusion from sequential optimization. Note that this command can only be called after `prototype_design`.

Synopsis

```
set_mem_dont_optimize [-observability] [-stability] [-stability_c]
    [-stability_s] [-sleep] [-clock_gate] [-all] [-reset]
    -inst <instance_list>
```

-observability

Specify `-observability` to exclude memories specified by `-inst` from memory gating during observability-based sequential optimization. Note that this option can only be specified before calling the `insert_mem_observability_gating` command.

-stability

Specify `-stability` to exclude memories specified by `-inst` from memory gating during stability-based sequential optimization (constant and symbolic). Note that this option can only be specified before calling the `insert_mem_stability_gating` command.

-stability_c

Specify `-stability_c` to exclude memories specified by `-inst` from memory gating during constant stability-based sequential optimization. Note that this option can only be specified before calling the `insert_mem_stability_gating -type C` command.

-stability_s

Specify `-stability_s` to exclude memories specified by `-inst` from memory gating during symbolic stability-based sequential optimization. Note that this option can only be specified before calling the `insert_mem_stability_gating -type S` command.

-sleep

Specify `-sleep` to exclude memories specified by `-inst` from memory gating light sleep-based sequential optimization. Note that this option can only be used with those memories that have a light sleep pin. Additionally, this option can only be specified before calling the `insert_mem_sleep_gating` command.

-clock_gate

Specify `-clock_gate` to exclude memories specified by `-inst` from memory clock gating optimization.

-all

Specify `-all` to exclude the memories specified by `-inst` from being gated during any type of sequential optimization.

-reset

Specify `-reset` to remove all types of dont-optimize constraints applied to memories specified by `-inst`. Note that this option overrides all other options.

-inst *<instance_list>* ...

A whitespace separated list of hierarchical instances or memories, for which the optimization constraints have been specified. For hierarchical instances, the optimization constraints are applied to all memories in the hierarchy.

Related Commands

insert_mem_observability_gating, insert_mem_sleep_gating, insert_mem_stability_gating, set_dont_optimize, set_mem_optimize, set_optimize

Usage

The **set_mem_dont_optimize** command sets the specified dont-optimize constraint on the given list of memory instances. For hierarchical instances, the dont-optimize constraint is applied to all memories in the specified hierarchy. When specified for some memories, only those memories are excluded from sequential optimization.

Examples

Example 1: Specifies that the memories `mem_inst1` and `I1.mem_inst2` be excluded from observability-based sequential optimization.

```
set_mem_dont_optimize -inst {mem_inst1 I1.mem_inst2} -observability
```

Example 2: Specifies that the memories `mem_inst1` and `I1.mem_inst2` be excluded from observability-based sequential optimization. However, since the second call applies the `-reset` option on the memory instance `mem_inst1`, the first call to exclude it from observability-based sequential optimization will be ignored. As a result, only `I1.mem_inst2` will be excluded from observability-based sequential optimization.

```
set_mem_dont_optimize -inst {mem_inst1 I1.mem_inst2} -observability
set_mem_dont_optimize -inst {mem_inst1} -reset
```

Example 3: Specifies that the memory `mem_inst1` be considered only for symbolic stability-based sequential optimization.

```
set_mem_dont_optimize -inst mem_inst1 -observability -stability_c
   -sleep -clock_gate
```

Example 4: Specifies that the memory instance `mem_inst` be excluded from memory clock-gating sequential optimization.

```
set_mem_dont_optimize -inst mem_inst -clock_gate
```

set_mem_optimize

set_mem_optimize — Marks memories for sequential optimization. Note that this command can only be called after `prototype_design`.

Synopsis

```
set_mem_optimize [ -observability] [ -stability] [ -stability_c]
    [ -stability_s] [ -sleep] [ -clock_gate] [ -all] [ -reset]
      -inst <instance_list>
```

-observability

Specify `-observability` to consider memories specified with `-inst` for gating during observability-based sequential optimization. Note that this option can only be called before the `insert_mem_observability_gating` command.

-stability

Specify `-stability` to consider memories specified with `-inst` for gating during constant and symbolic stability-based sequential optimization. Note that this option can only be called before the `insert_mem_stability_gating` command.

-stability_c

Specify `-stability_c` to consider memories specified with `-inst` for gating during constant stability-based sequential optimization. Note that this option can only be called before the `insert_mem_stability_gating -type C` command.

-stability_s

Specify `-stability_s` to consider memories specified with `-inst` for gating during symbolic stability-based sequential optimization. Note that this option can only be called before the `insert_mem_stability_gating -type S` command.

-sleep

Specify `-sleep` to consider memories specified with `-inst` for gating during during light sleep-based sequential optimization. Note that this option can only be specified for those memories that have a light sleep pin. Additionally, this option can only be called before calling the `insert_mem_sleep_gating` command.

-clock_gate

Specifies that the memories be considered for gating during memory clock gating optimization.

-all

Specifies that all memories in the design be considered for all all types of sequential optimizations.

-reset

Removes optimization constraints applied on the memories.

-inst `<instance_list>`

A whitespace separated list of hierarchical instances or memories, for which the optimization constraints have been specified. For hierarchical instances, the optimization constraints are applied to all memories in the hierarchy.

Related Commands

insert_memory_clock_gating, insert_mem_observability_gating, insert_mem_sleep_gating, insert_mem_stability_gating, set_dont_optimize, set_mem_dont_optimize, set_optimize

Usage

The **set_mem_optimize** command sets optimization constraints of the specified type on the given list of hierarchical instances or memories. For hierarchical instances, the optimize constraint is applied to all memories in the hierarchy. When this command is not specified, all memories in the design are available for sequential optimization. When specified for some memories, only those memories are available for sequential optimization.

Examples

Example 1: Specifies that the memory instance `mem_inst1` be gated during observability-based sequential optimization. All the other memories are not optimized.

```
set_mem_optimize  -inst mem_inst1  -observability
```

Example 2: Specifies that the memory instance `mem_inst1` be optimized for observability-based sequential optimization. However, the second call removes all constraints applied on `mem_inst1`. As a result, all memories would now be available for all types of sequential optimizations.

```
set_mem_optimize -inst mem_inst1 -observability
set_mem_optimize -inst mem_inst1 -reset
```

Example 3: Specifies that the memory instance `mem_inst1` be considered for observability, constant stability and sleep based sequential optimizations. All other memories are not considered.

```
set_mem_optimize -inst mem_inst1 -observability -stability_c -sleep
```

Example 4: Specifies that the memory instance `mem_inst` be considered for memory clock -gating sequential optimization only. All other memories are not considered for any type of sequential optimization.

```
set_mem_optimize -inst mem_inst -clock_gate
```

set_mem_override

set_mem_override — Sets the override port/signal to disable memory gating and light sleep gating. Note that this command can only be called after `build_design` and before any sequential optimization command.

Synopsis

```
set_mem_override { -port <port_name> | -signal <signal_name> }
    { -mg |  -ls |  -all } [ -auto ] [ -target_clock <clock_name> ]
    [ -target_polarity <pos | neg> ]
    [ -mem_list <memory_list> |  -inst_list <instance_list> ]
    [ -ignore_transition]
```

-port *<port_name>*

Specifies the name of port to be used as the override port. If the port is not present in the design and the `-auto` option is not specified, PowerPro exits with an error. To set an existing port as the override port, it must either be a top hierarchy input port or an output port other than the top hierarchy output port.

-signal *<signal name>*

Specifies the name of signal to be used as the override signal. If the signal is not present in the design, PowerPro exits with an error. To set an existing signal as the override signal, it must be driven by a top hierarchy input port or an output port other than the top hierarchy output port.

-mg | -ls | -all

Specify `-mg` to override memory gating hierarchies only.

Specify `-ls` to override light sleep gating hierarchies only.

Specify `-all` to override both memory gating and light sleep gating hierarchies.

-auto

Specify `-auto` to automatically create the override port on the top -level, in case it does not exist in the design. Note that this option requires the port name to be specified using `-port`.

-target_clock *<clock_name>*

Specifies that the port/signal be used as an override port/signal only for those memories that are driven by *<clock_name>*.

-target_polarity pos | neg

Specifies that the port/signal be used as an override port/signal only for those memories whose clock network is of the mentioned polarity. Note that this option requires the clock name to be specified using the `-target_clock` option.

-mem_list *<memory_list>* | -inst_list *<instance_list>*

Specify `-mem_list` to provide a list of memory instances to override.

Specify `-inst_list` to provide a list of instances, for which all the memory instances within are to be overridden with the specified override port/signal.

In case neither of these options is specified, all memories will be overridden with the specified override port/signal.

-ignore_transition

Specifies that the override port/signal will not transition during the run.

Related Commands

insert_mem_observability_gating, insert_mem_sleep_gating, insert_mem_stability_gating

Usage

The **set_mem_override** command is used to specify the override port/signal in the design. The override port/signal is used to disable memory gating logic added by PowerPro. The memory gating logic can be disabled by setting the override port/signal to 1 in the RTL.

Multiple `set_mem_override` commands can be issued to override different memories with different override ports/signals. In case multiple calls are specified for the same memory, the last call prevails and overwrites any settings made by the previous call.

Examples

Example 1: Specifies that the port `override1` be used as override port for all memory gating hierarchies in the design and that the port `override2` be used as the override port for all sleep gating hierarchies in the design. The use of the `-ignore_transition` option indicates that the override ports `override1` and `override2` will not transition during the execution of the chip. In case these override ports do not exist in the design, an error will be issued.

```
set_mem_override -mg -port override1 -ignore_transition
set_mem_override -ls -port override2 -ignore_transition
```

Example 2: Specifies that the port `override` be used as the override port for the memory gating hierarchies of memories `a.m1` and `b.m2`. The rest of the design memories remain untouched. In case the port `override` does not exist, a port by this name is automatically created on the top hierarchy.

```
set_mem_override -mg -port override -auto -mem_list "a.m1 b.m2"  \
       -ignore_transition
```

Example 3: Specifies that the port `override` be used as the override port for the sleep gating hierarchies within the instance `a.b`.

```
set_mem_override -ls  -port override -auto -inst_list "a.b"  \
       -ignore_transition
```

set_mode_override

set_mode_override — Specifies the override strategy to be used by PowerPro for a given mode. Note that this command can only be called after `build_design` and before `prototype_design`.

Synopsis

```
set_mode_override  -mode <mode_name>
    { -port <port_name> | -signal <signal_name> | -infer }
    [ -auto_create_port ]
```

-mode *<mode_name>*

Specifies the name of the mode to be associated with this override strategy. If the mode does not exist, PowerPro exits with an error. To create a mode, use the `create_mode` command.

-port *<port_name>* | -signal *<signal_name>* | -infer

Specify `-port` to use a port to override the PowerPro generated logic. When specified, PowerPro does not apply additional override logic based on mode constraints.

Specify `-signal` to use a signal to override the PowerPro generated logic. When specified, PowerPro does not apply additional override logic based on mode constraints.

Specify `-infer` to instruct PowerPro to infer and create the mode override from the mode constraints defined using the `create_mode_constraint` command. This is the default behavior when `set_mode_override` is not explicitly specified.

-auto_create_port

Automatically creates the port specified with the `-port` option when the port does not exist. The override port is created on the top hierarchy.

Note that this option is honored only when the `-port` option is specified.

Related Commands

create_mode, create_mode_constraint

Usage

The **set_mode_override** command is used to specify the override strategy to be used by PowerPro for a given mode. In case the mode does not already exist, use the `create_mode` command to create the mode before specifying this command.

Examples

Example 1: Sets the default override strategy for the mode `awake`. Accordingly, the override logic is generated using the mode constraints of the mode `awake`.

```
set_mode_override -mode awake  -infer
```

Example 2: Sets `mode_ov` as the override strategy for the mode `sleep`. Here the `mode_ov` port is directly used as the override strategy. In case the port `mode_ov` does not exist in the design, PowerPro exits with an error.

```
set_mode_override -mode sleep  -port mode_ov
```

In the same example, to prevent PowerPro from exiting if the port `mode_ov` does not exist, specify the `-auto_create_port` option as follows:

```
set_mode_override -mode sleep  -port mode_ov  -auto_create_port
```

By adding the `-auto_create_port` option, a new override port by the name of `mode_ov` is created on the top hierarchy in case the port does not already exist.

set_module_as_cgic

Deprecated w.e.f. PowerPro 10.4.

set_module_as_cgic — Specifies the CGIC modules to be used for memory clock gating.

Synopsis

```
set_module_as_cgic  -module <mod_name>
     -clock<cgic_clock>
     -clock_gate_enable_pin <cgic_enable>
     -clock_gate_out_pin <cgic_output>
     -clock_gate_test_pin <cgic_test_pin>
```

-module *<mod_name>*

Specifies the CGIC module to be used for clock gating memories specified using the `set_cell_as_memory` command.

The CGIC module *<mod_name>* can be a module used in the design or a CGIC cell read in the constraints library using the command `build_design -cons` command.

Note that the module should have been compiled, otherwise `set_module_as_cgic` will generate an error.

-clock *<cgic_clock>*

Specifies the name of the clock port for the clock gating module. Note that this port must exist in the module, else PowerPro will error out.

-clock_gate_enable_pin *<cgic_enable>*

Specifies the name of the clock gating enable port. Note that this port must exist in the module, else PowerPro will error out.

-clock_gate_out_pin *<cgic_output>*

Specifies the name of the output port for the clock gating module. Note that this port must exist in the module, else PowerPro will error out.

-clock_gate_test_pin *<cgic_test_pin>*

Specifies the name of the scan enable port for the clock gating module. Note that this port must exist in the module, else PowerPro will error out.

Related Commands

set_scan_enable, set_cell_as_memory, set_target_cgic

Usage

The `set_module_as_cgic` command is used to specify the CGIC modules to be used for memory clock gating.

Notes:

- This command must be executed soon after `prototype_design` in the Memory Clock Gating flow.
- In case a port is not connected, it will be connected to `0`.

Examples

Example 1: In this example, the CGIC cell `my_CG_MOD` is used to clock gate the memory `p1024x32m_inst2`.

```
build_design design.v memory.v
build_design  -cons cgic.v
create_clock  -name CLK  -period 123.45 clk
prototype_design

set_module_as_cgic  -module my_CG_MOD  -clock ck_in
-clock_gate_enable_pin enable  -clock_gate_out_pin ck_out
-clock_gate_test_pin test

set_scan_enable  -default  -signal SCAN_EN

set_cell_as_memory  -module p1024x32m  -enable_polarity_clock "me pos
clk"  -clock_and_polarity "clk pos"
```

set_multi_vth_constraint

set_multi_vth_constraint — Sets `multi_vth` constraints for the design. Note that this command can only be called after `build_design` and before `prototype_design`.

Synopsis

```
set_multi_vth_constraint {-vth_group <list_of_group_names> }
    {-vth_percentage <percentage> } {-vth_type <threshold_type> }
    [-instance <instance_name> ]
```

-vth_group *<list_of_group_names>*

Specifies the names of the groups that must be considered to be a particular threshold group at the specified instance.

-vth_percentage *<percentage>*

Specifies the percentage of distribution that this particular threshold group should acquire. Percentage is an integer value between `0` and `100`.

-vth_type *<threshold_type>*

Specifies the threshold type of the threshold group. These can be of three types, that is, `lvt`, `svt`, and `hvt`. If `lvt` is used, the tool must know that this group should be considered as a low threshold voltage group for all the internal purposes. Similarly, `svt` is used for standard threshold voltage group, and `hvt` is used for high threshold voltage group.

-instance *<instance_name>*

If this option is specified, the constraint is applied only to the instance specified by *<instance_name>*. If not specified, the constraint is applied to the top hierarchy.

-lvth_group

The `-lvth_group` option is marked for deprecation in a future release. Use the option `-vth_group` as the replacement for the `-lvth_group` option.

-lvth_percentage

The `-lvth_percentage` option is marked for deprecation in a future release. Use the option `-vth_percentage` as the replacement for the `-lvth_percentage` option.

Note: The distribution of the register instances happens only in the spef-less flow.

Related Commands

set_default_threshold_voltage_group, set_threshold_voltage_group

Usage

The **set_multi_vth_constraint** command sets `multi_vth` constraints for the design.

A sample script is shown below.

```
# Reads-in the library lib1
read_library file1.lib

# Reads-in the library lib2
read_library file2.lib

# Specifies that all cells in the library lib1 which do not
# have the threshold_voltage_group attribute specified
# be assigned to the group DEFAULT.
set_default_threshold_voltage_group -name "DEFAULT" -library lib1

# Specifies that all cells with a name starting with LVT_, in the
# library lib1, be assigned to the group LOW1.
set_threshold_voltage_group -name "LOW1" -cells [find_lib_cell –
wildcard –cell lib1/LVT_*]

# Specifies that all cells in the library lib2 which do not
# have the threshold_voltage_group attribute specified be
# assigned to the group LVTH_DEF
set_default_threshold_voltage_group -name "LVTH_DEF" -library lib2

# Specifies that all cells with a name prefixed with SVT_,
# in the library lib2, be assigned to the group LOW2
set_threshold_voltage_group -name "LOW2" -cells [find_lib_cell –
wildcard –cell lib2/SVT_*]

# Specifies that all cells with a name prefixed with HVT_,
# in the library lib2, be assigned to the group HIGH
set_threshold_voltage_group -name "HIGH" –cells [find_lib_cell –
wildcard -cell lib2/HVT_*]

# Specifies that all cells belonging to the group LOW1 and LVTH_DEF
# i.e., LVT_* cells from lib1 and all cells except those prefixed
# with SVT_ and HVT_ from the library lib2 be considered as
# low vth cells and their percentage in the design should be
approximately 20%
set_multi_vth_constraint -vth_group "LOW1 LVTH_DEF" -vth_percentage 20
-vth_type “lvt”

# Specifies that all cells belonging to the group LOW2
# i.e., SVT_ cells from lib2 be considered as
# standard threshold cells and their percentage in the design should be
approximately 50%
set_multi_vth_constraint -vth_group  "LOW2" -vth_percentage 50 -
vth_type “svt”
# Specifies that all cells belonging to the group HIGH
# i.e., HVT_ cells from lib2 be considered as
# high threshold cells and their percentage in the design should be
approximately 30%
set_multi_vth_constraint -vth_group "HIGH" -vth_percentage 30 -vth_type
“hvt”
...
prototype_design
...
```

Note: The names `DEFAULT`, `LOW1`, `LOW2`, `HIGH` and `LVTH_DEF` are not legal names and do not have any significance. They can be replaced by any string.

Examples

Example 1: Consider the following library files.

```
//File: file1.lib

//All cells in lib1 do not have the threshold_voltage_group attribute
specified
library(lib1)
{
...
   default_threshold_voltage_group : "lvt" ;
...

// File: file2.lib

library(lib2)
{
...

   default_threshold_voltage_group : "high_threshold" ;
...
   cell(lvt_and) {
      threshold_voltage_group : "low_threshold" ;
     ....
   }
   cell(lvt_inv) {
      threshold_voltage_group : "low_threshold" ;
     ....
   }

//All svt_* cells have the threshold_voltage_group attribute specified.

   cell(svt_and) {
      threshold_voltage_group : "standard_threshold" ;
     ....
   }
   cell(svt_inv) {
      threshold_voltage_group : "standard_threshold" ;
     ....
   }

//All hvt_* cells do not have the threshold_voltage_group attribute
specified.
//They belong to the default threshold voltage group named
"high_threshold".

   cell(hvt_and) {
     ....
   }
   cell(hvt_inv) {
     ....
   }
```

Assuming the above mentioned libraries have been read, specifying the following command instructs PowerPro to map 20% of the combinational and register instances to the `low-vth` groups specified, 50% of the combinational and register instances to the `standard-vth` group specified, and 30% of the combinational and register instances to the `high-vth` group specified.

```
read_library file1.lib
read_library file2.lib
...
set_multi_vth_constraint -vth_group "lvt low_threshold"  -
vth_percentage 20 -vth_type “lvt”
set_multi_vth_constraint -vth_group "standard_threshold" -
vth_percentage 50 -vth_type “svt”
set_multi_vth_constraint -vth_group "high_threshold" -vth_percentage 30
-vth_type “hvt”
...
```

In this case:

- The first `set_multi_vth_constraint` command applies constraints on all combinational and register instances with the library cells belonging to the `lvt group` (i.e. all cells in `lib1`) and all cells belonging to the `low_threshold` group (i.e. cell names prefixed with `lvt_` in `lib2`).
- The second `set_multi_vth_constraint` command applies constraints on all combinational and register instances with the library cells belonging to the `standard_threshold` group (i.e. cell names prefixed with `svt_` in `lib2`).
- The third `set_multi_vth_constraint` command applies constraints on all combinational and register instances with the library cells belonging to the `high_threshold` group (i.e. cells which do not have `threshold_voltage_group` specified in `lib2`, thus belonging to the default `high_threshold` group).

set_observable

set_observable — Sets a signal or port to be always observable during observability-based sequential optimization. Note that this command can only be called after `build_design`.

Synopsis

`set_observable` *`<list_of_signals_or_ports>`*`...`

`<list_of_signals_or_ports>`...

A whitespace separated list of signals or ports to be marked as always observable during observability-based sequential optimization.

Related Commands

insert_observability_logic

Usage

When a port or signal is marked observable using the **set_observable** command, observability dont-care conditions are not propagated across them.

Examples

Example 1: Marks the port `in2` of the module as observable during observability-based sequential optimization.

```
set_observable in2
```

set_operating_condition

set_operating_condition — Specifies the operating condition for an instance or design. Note that the technology libraries must have been read using the `read_library` command prior to calling this command. Additionally, the `set_operating_condition` command can only be called before `prototype_design`.

Synopsis

```
set_operating_condition <op_cond_name>  -library <lib_name>
     -inst <instance_list>  -list
```

`<op_cond_name>`

Specifies the operating condition to be applied for power estimation. Note that this condition must exist in the library specified.

-library *`<lib_name>`*

Specifies the name of the technology library containing the definition of the operating condition. This library must have been read using the `read_library` command.

-inst *`<instance_list>`*

Specifies a list of hierarchical names for instances on which the operating condition is to be applied. If not specified, the operating condition is applied to all design objects.

-list

Displays a list of operating conditions in the design. When this option is specified, all other options are ignored.

Related Commands

report_power, set_binding, set_target_technology

Usage

The **set_operating_condition** command allows one to specify a different set of operating conditions for power estimation. By default, PowerPro performs power estimation using `default_operating_conditions` of the first technology library specified using `set_target_technology`. If the `set_target_technology` command has not been specified, the `default_operating_conditions` of the first technology library read using `read_library` is used.

To get a list of all operating conditions, use the `-list` option. Note that when this option is specified, the `set_operating_condition` command only displays a list of operating conditions and does not apply any constraints.

If `operating_conditions` is not specified for a hierarchical instance, it is derived from the parent hierarchy. Consider the following design.

```
//File: file1.lib

library(lib1) //contains opcond1 and opcond2
{
...
  operating_conditions(opcond1) {
    process : 1;
    temperature :  -40;
    voltage : 1.32;
  }
  operating_conditions(opcond2) {
    process : 1.36;
    temperature : 25.0;
    voltage : 1.200;
  }
...

//File: file2.lib

library(lib2) //contains opcond1
{
...
  operating_conditions(opcond1) {
    process : 1.00;
    temperature : 125.0;
    voltage : 1.200;
  }
...

//File: top.v
module inner_level1
   inner_level2 f2(...);
   ...
endmodule

module top
   inner_level1 f1 ( ..);
   ...
endmodule
```

In this example, the operating condition opcond1 in the library `lib2` will be applied to the design objects in `f1.f2`. For the remaining design objects within `f1`, the operating condition `opcond2` defined in the library `lib1` will be used. For all other design objects, the operating condition `opcond1` specified in library `lib1` will be applied.

```
...
read_library file1.lib  //contains lib1
read_library file2.lib  //contains lib2
...
set_operating_condition opcond2  -library lib1  -inst f1
set_operating_condition opcond1 -library lib2 -inst f1.f2
set_operating_condition opcond1 -library lib1
```

Example

Example 1: Applies the operating condition `opcond2`, defined in the library `lib1`, to the instance `f1`.

```
....
read_library file1.lib  //contains lib1
....
set_operating_condition opcond2 -library lib1 -inst f1
```

Example 2: Applies the operating condition `opcond2`, defined in the library `lib3`, to the instance `f1` and `f2`. For the remaining design objects, the `default_operating_conditions` of the first library specified by `set_target_technology` i.e., `lib2` applies.

```
....
read_library file1.lib  //contains lib1
read_library file2.lib  //contains lib2
read_library file3.lib  //contains lib3
....
set_target_technology  -library "lib2 lib3"
....
set_operating_condition opcond2  -library lib3  -inst "f1 f2"
```

set_optimize

set_optimize — Marks flops for optimization. Note that this command can only be called after `prototype_design`.

Synopsis

```
set_optimize [ -observability ] [ -stability ] [ -stability_c ]
     [ -stability_s ] [ -all ] [ -auto_avoiding ] [ -inst
    <instance_list> ]
```

-observability

Specify `-observability` to mark the list of hierarchical instances or flops specified by `-inst` for observability-based sequential optimization only. Note that this option can only be called before calling the `insert_observability_logic` command.

-stability

Specify `-stability` to mark the list of hierarchical instances or flops specified by `-inst` for stability-based sequential optimization only (constant and symbolic). Note that this option can only be called before calling the `insert_stability_logic` command.

-stability_c

Specify `-stability_c` to mark the list of hierarchical instances or flops specified by `-inst` for constant stability-based sequential optimization only. Note that this option can only be called before calling the `insert_stability_logic -type C` command.

-stability_s

Specify `-stability_s` to mark the list of hierarchical instances or flops specified by `-inst` for symbolic stability-based sequential optimization only. Note that this option can only be called before calling the `insert_stability_logic -type S` command.

-all

Specify `-all` to mark the list of hierarchical instances or flops specified by `-inst` for both observability and stability-based sequential optimization.

-auto_avoiding

Specify `-auto_avoiding` to skip the hierarchical instances or flops specified by `-inst` (that are already set as don't_optimize) for both observability- and stability-based sequential optimizations.

-inst *<instance_list>*

A whitespace separated list of hierarchical instances or flops marked for optimization. All other flops are excluded from any type of sequential optimization. For hierarchical instances, the optimization constraint is applied to all flops in the hierarchy.

Related Commands

set_dont_optimize, set_mem_dont_optimize, set_mem_optimize, insert_observability_logic, insert_stability_logic

Usage

The **set_optimize** command marks a list of instances for a specific type of sequential optimization. All other instances in the design are excluded from any type of sequential optimization.

Examples

Example 1: Marks all flops in the design for observability and stability-based sequential optimization.

```
prototype_design
...
set_optimize
...
insert_observability_logic
insert_stability_logic
```

Example 2: Marks the flops `M1.ff1` and `M1.ff2` for observability-based sequential optimization. All other flops in the design are not optimized.

```
set_optimize  -inst "M1.ff1 M1.ff2"  -observability
```

Example 3: Marks the flops `M1.ff1` and `M1.ff2` for both observability-based and stability-based sequential optimization. All other flops in the design are left as-is.

```
set_optimize  -inst "M1.ff1 M1.ff2"  -all
```

Example 4: Skips the hierarchical instances or flops that are already set as dont_optimize.

```
set_optimize -inst "M1.ff1 M1.ff2" -auto_avoiding
```

set_override_stability

set_override_stability — Ignores the stability condition across the specified flop or signal. Note that this command can be called only after the `prototype_design` command.

Synopsis

```
set_override_stability [ -const |  -sym |  -all ]
    { -signal <signal_name> |  -flop <flop_name> }
```

-const | -sym | -all

Specifies the type of constraint to ignore.

Specify `-const` to override constant stability conditions across the specified flop or signal.

Specify `-sym` to override symbolic stability conditions across the specified flop or signal.

Specify `-all` to override both constant stability and symbolic stability conditions for the flop or signal specified. This is also the default.

-signal *<signal_name>* | -flop *<flop_name>*

Specifies the hierarchical name of the signal or flop for which the stability condition is to be ignored.

Related Commands

insert_stability_logic

Usage

Use the **set_override_stability** command to ignore a stability condition specified for a flop or signal.

Example

Example 1: Ignores the stability condition across the flop `f1`.

```
build_design input.v
…
prototype_design
…
set_override_stability  -flop f1
insert_stability_logic
```

set_powerpro_mode

set_powerpro_mode — Sets the PowerPro operating mode.

Synopsis

`set_powerpro_mode` { eco }

eco

Specifies the PowerPro operating mode to be set.

Specify `eco` to prepare PowerPro for an ECO run. In this mode, PowerPro generates data which allows it to apply ECO (Engineering Change Order) changes in the automated power optimization flow. In the ECO mode, PowerPro requires that the following remain the same for the original run and the ECO run: constraints applied, clock and reset network, and UPF file.

The `guided` option has been deprecated. Use the `generate_guidance` command instead.

Related Commands

analyze_mem_sim_traces, report_data_gating, report_power, report_memory_redundant_access, report_redundant_write_info

Usage

The **set_powerpro_mode** command sets the PowerPro operating mode, which can be set for the Engineering Change Order flow (`eco`).

Note: Except for variable and global settings, this must be the first command specified. Specifying this command later in the flow leads to an error.

Example

Example 1: Sets the PowerPro operating mode for the ECO flow.

```
set_powerpro_mode eco
```

set_powerpro_reset

set_powerpro_reset — Specifies the signals to be used to initialize or reset the PowerPro generated sequential elements that are used in the stability sequential gating moves generated by PowerPro. Note that this command can only be called after `prototype_design` and before calling any sequential optimization command.

Synopsis

```
set_powerpro_reset  -target_clock <named_clock>
    [ -target_polarity <pos | neg>] [ -synchronizer_needed <yes | no> ]
    [ -default] [ -inst <hier_inst_path_to_block> ]
    [ -active_high |  -active_low ]  [ -sync |  -async ]
    [<list_of_signals>] [ -auto ]
    [ -synchronizer_module <sync_mod_name> ]
```

-target_clock *<named_clock>*

Specifies the name of the clock.

-target_polarity *<pos | neg>*

Specifies the polarity of the clock.

-synchronizer_needed *<yes | no>*

Specifies whether or not a synchronizer should be created when the clock of the reset signal differs from the clock for which the reset was queried.

-default

Marks the specified PowerPro reset as the default reset. Note that only one specification can be marked as the default. If new PowerPro resets are required for unspecified clock domains, tool would synchronize the signal(s) of this PowerPro reset for use with the other clock domain.

Note the `-target_clock` and `-target_polarity` options cannot be specified with the `-default` option.

-inst *<hier_inst_path_to_block>*

Specifies that the reset only applies to PowerPro flops contained within the specified instance and its child instances.

-active_high | -active_low

Specifies the active state of the reset signal.

-sync | -async

Specifies whether the reset needs to be applied synchronously or asynchronously. By default, resets are applied synchronously to the newly added registers.

<list_of_signals>

Specifies the list of reset signals. Signals should be associated with a valid clock domain. If the signal is not found in the design and the `-auto` option is not specified, the command will error out. If the signal is not found in the design and the `-auto` option is specified, a top -level port will be created.

-auto

Specifies that a new top level port be created if the specified reset port or signal does not exist.

-synchronizer_module *<sync_mod_name>*

Specifies the synchronizer module to be associated with the reset logic. Note that the synchronizer must have been specified using the `set_user_synchronizer` command. This synchronizer will be used to synchronize the reset signal w.r.t the target -clock when the clock domain of the reset signal differs from the clock domain of the target clock.

Related Commands

insert_stability_logic, report_clocks

Usage

The **set_powerpro_reset** command is used to specify reset signals for PowerPro-generated sequential elements before any command that may generate such elements is invoked. In particular, the `insert_stability_logic` and `insert_mem_stability_gating` commands perform stability optimization, and normally generate flops with reset ports.

If a reset is required for the PowerPro-generated sequential element and the `set_powerpro_reset` constraint is not provided, PowerPro adds a new reset signal in the design and issues the following command:

```
set_powerpro_reset -default -auto -active_low POWERPRO_RESET_N
```

It is important that the post-optimization design be reset correctly using a reset signal(s) with appropriate edge sensitivity to avoid any functional errors or mismatches with the test vectors.

Examples

Example 1: In this example, `rst1` and `rst2` are specified as resets for the PowerPro generated flops that lie in clock domain `clk` with positive clock phase. Because both `rst1` and `rst2` are active high signals, if any of these signals assume the value of 1, the flops in the positive clock domain `clk` will be reset.

```
set_powerpro_reset  -target_clock clk  -target_polarity pos
-active_high "rst1 rst2"
```

Example 2: In this example, `rst1` and `rst2` are specified as resets for the PowerPro generated flops that lie in clock domain `clk` with negative clock phase. Because both `rst1` and `rst2` are active high signals, if any of these signals assume the value of 1, the flops in the negative clock domain `clk` will be reset.

```
set_powerpro_reset  -target_clock clk  -target_polarity neg
-active_high "rst1 rst2"
```

Example 3: In this example, `rst1` and `rst2` are specified as resets for the PowerPro generated flops that lie in clock domain `clk` with negative clock phase. If `rst1` or `rst2` does not exist in the design, a top level port is created.

Because both `rst1` and `rst2` are active low signals, if any of these signals assume the value of 0, the flops in the negative clock domain `clk` will be reset.

```
set_powerpro_reset  -target_clock clk  -target_polarity neg
-active_low "rst1 rst2"  -auto
```

Example 4: In this example, the PowerPro reset is set as the default reset. If for any clock domain, `set_powerpro_reset` is not specified then reset signals of default `set_powerpro_reset` (`rst1 and rst2`) are used as reset signals.

```
set_powerpro_reset  -active_low "rst1 rst2"  -default
```

Example 5: In this example, `rst1` and `rst2` are specified as resets for PowerPro -generated flops that lie in the clock domain `clk` with positive clock phase. This reset logic is applicable to the flops that lie within the instance `inst1` and its sub -instances.

```
set_powerpro_reset  -target_clock clk  -target_polarity pos
-active_low "rst1 rst2"  -inst "inst1"
```

Example 6: In this example, `rst1` and `rst2` are set as resets for PowerPro -generated flops that lie in the clock domain `clk` with positive clock phase. This reset logic is applicable to the flops that lie within the instance `inst1` and its sub instances.

Note that if the clock domain of `rst1` or `rst2` differs from `clk` then synchronizer `sync_mod` will be used to synchronize that signal.

```
set_powerpro_reset  -target_clock clk  -target_polarity pos
-active_low "rst1 rst2"  -inst "inst1"  -synchronizer_module sync_mod
```

set_rtl_to_gate_name

set_rtl_to_gate_name — Specifies the RTL name corresponding to the gate-level netlist element. This RTL name is used in the `read_fsdb` command if the global `pa_rtl_sim_on_glpa` is enabled. To run the `set_rtl_to_gate_name` command, the name of the file that contains a list of the `set_rtl_to_gate_name` commands must be passed as an argument to the `read_name_map_file` command.

Synopsis

```
set_rtl_to_gate_name {-rtl <rtl_name>} {-gate <gate_name>} [-inverted]
```

-rtl <*rtl_name*>

Specifies the RTL name.

-gate <*gate_name*>

Specifies the name of the gate-level netlist element.

-inverted

Specifies if the gate-level netlist element name (specified by `gate_name`) is logically invert of the RTL element (specified by `rtl_name`).

Related Commands

read_name_map_file

Usage

Specifies the RTL name corresponding to the gate-level netlist element. This RTL name is used in the `read_fsdb` command if the global `pa_rtl_sim_on_glpa` is enabled. To run the `set_rtl_to_gate_name` command, the name of the file that contains a list of the `set_rtl_to_gate_name` commands must be passed as an argument to the `read_name_map_file` command.

Examples

Example 1: Specifies the RTL name to gate-level netlist element name.

```
set_rtl_to_gate_name -rtl {r_out1[2][5]} -gate {r_out1_2__5_}
```

Example 2: Specifies the RTL name to gate-level netlist element name using the –`inverted` option.

```
set_rtl_to_gate_name -rtl {A/B/C/xyz[2]} -gate {A_B/C/xyz_reg_2} -
inverted
```

set_sa

set_sa — Annotates a port or signal with switching activity information. Note that this command can only be called after `build_design` and before `prototype_design`.

Synopsis

```
set_sa {  -port <port_name> |  -signal <signal_name> }
    {  -td <value> |  -prob <value> }
```

-port *<port_name>* | -signal *<signal_name>*

Specifies the type of object to be annotated with the switching activity information, where the name specified must be the full hierarchical name.

Specify -`port` to annotate a port with switching activity information.

Specify -`signal` to annotate a signal with switching activity information.

-td *<value>* | -prob *<value>*

Specifies the type of switching activity data, where *<value>* is a floating point value greater than or equal to 0.

Specify -`prob` to indicate that a probability value is to be annotated on the signal/port, where the probability value indicates the percentage of time the signal or port assumes the logic value 1. So, specifying .2 as the value would mean that the signal/port assumes the value 1, 20% of the time. Value must be a floating point number in the range [0,1].

Specify -`td` to indicate that a toggle density value is to be annotated on the port/signal, where toggle density indicates the number of times a signal or port toggles its value per unit time. Value must be a floating point number greater than or equal to `0`.

Related Commands

get_sa, read_fsdb, read_saif

Usage

The **set_sa** command allows one to set switching activity data for the specified signal or port. This switching activity data may be Probability or Toggle Density.

Examples

Example 1: Annotates switching activity information to the 3-bit signal `A`. Assuming the following commands are specified:

```
set_sa  -signal A[0]  -td 0.004
set_sa  -signal A[1]  -td 0.001
set_sa  -signal A[2]  -td 0.002
```

The results would be as follows:

- After the first run, all 3 bits are assigned the value `0.004`. Any earlier TD values assigned to the signal `A` are lost.
- After the second run, as TD on `A` is user asserted, only `A[1]` is assigned `0.001`. So, `A[0]` and `A[2]` continue to have the value `0.0004`.
- After the third run, as TD on `A` is user asserted, only `A[2]` is assigned `0.002`.
- End result: `A[0] = 0.004, A[1] = 0.001, and A[2] = 0.002.`

set_scaling

set_scaling — Specifies the constraints that can be used to scale the different parameters of power. These parameters are used to compute the design power.

Note that this command can be called only after the `build_design` command and before the `prototype_design` command.

Synopsis

```
set_scaling [-inst <instance_name>]
 {[-flop_switching_cap <cap_scaling_factor>]
  [-flop_leakage_power <power_scaling_factor>]
  [-flop_internal_power <power_scaling_factor>]
  [-flop_switching_power <power_scaling_factor>]
  [-flop_cell_sizing_level <flop_cell_sizing_level>]
  [-memory_switching_cap <cap_scaling_factor>]
  [-memory_leakage_power <power_scaling_factor>]
  [-memory_internal_power <power_scaling_factor>]
  [-memory_switching_power <power_scaling_factor>]
  [-clocktree_switching_cap <cap_scaling_factor>]
  [-clocktree_leakage_power <power_scaling_factor>]
  [-clocktree_internal_power <power_scaling_factor>]
  [-clocktree_switching_power <power_scaling_factor>]
  [-clocktree_buffer_sizing_level <clock_buffer_sizing_level>]
  [-comb_switching_cap <cap_scaling_factor>]
  [-comb_leakage_power <power_scaling_factor>]
  [-comb_internal_power <power_scaling_factor>]
  [-comb_switching_power <power_scaling_factor>]
  [-comb_glitch_derating <glitch_derating_factor>]
  [-comb_frequency <freq_scaling_factor>]
  [-comb_cell_sizing_level <comb_cell_sizing_level>]}
```

-inst <*instance_name*>

Applies the constraint on the specified instance. The instance can be a hierarchical or a primitive instance.

If the constraint is applied on a hierarchical instance, it is automatically applied on all the sub-hierarchical instances that do not have the same constraint applied on them.

Note: If this option is not specified, the constraint is applied on the top hierarchy.

-flop_switching_cap <*cap_scaling_factor*>

Specifies the factor by which the capacitance of the flop-driven signals must be scaled for the specified instance. The data type of the scaling factor is `float`.

-flop_leakage_power <*power_scaling_factor*>

Specifies the factor by which leakage power of the flop must be scaled for the specified instance. The data type of the scaling factor is `float`.

-flop_internal_power <*power_scaling_factor*>

Specifies the factor by which the internal power of the flop must be scaled for the specified instance. The data type of the scaling factor is `float`.

-flop_switching_power <*power_scaling_factor*>

Specifies the factor by which the switching power of the flop must be scaled for the specified instance. The data type of the scaling factor is `float`.

-flop_cell_sizing_level <*flop_cell_sizing_level*>

Specifies the level by which the register elements must be upsized in the mapped netlist for the specified hierarchical instance. The criteria for upsizing is the cell area and average leakage power. This scaling can only be called on hierarchical instances. This option is honored only in SPEF-less flow. The data type of the scaling factor is `integer`.

-memory_switching_cap <*cap_scaling_factor*>

Specifies the factor by which the capacitance of the memory-driven signals must be scaled for the specified instance. The data type of the scaling factor is `float`.

-memory_leakage_power <*power_scaling_factor*>

Specifies the factor by which the leakage power of the memory must be scaled for the specified instance. The data type of the scaling factor is `float`.

-memory_internal_power <*power_scaling_factor*>

Specifies the factor by which the internal power of the memory must be scaled for the specified instance. The data type of the scaling factor is `float`.

-memory_switching_power <*`power_scaling_factor`*>

Specifies the factor by which the switching power of the memory must be scaled for the specified instance. The data type of the scaling factor is `float`.

-clocktree_switching_cap <*`cap_scaling_factor`*>

Specifies the factor by which the signal capacitance of the clock network must be scaled for the specified instance. The data type of the scaling factor is `float`.

- clocktree_leakage_power <*`power_scaling_factor`*>

Specifies the factor by which the leakage power of the clock network must be scaled for the specified instance. The data type of the scaling factor is `float`.

- clocktree_internal_power <*`power_scaling_factor`*>

Specifies the factor by which the internal power of the clock network must be scaled for the specified instance. The data type of the scaling factor is `float`.

- clocktree_switching_power <*`power_scaling_factor`*>

Specifies the factor by which the switching power of the clock network must be scaled for the specified instance. The data type of the scaling factor is `float`.

- clocktree_buffer_sizing_level <*`clock_buffer_sizing_level`*>

Specifies the level by which the buffers in the clock tree must be upsized or downsized during the clock-tree synthesis. The data type of the scaling factor is `integer`.

-comb_switching_cap <*`cap_scaling_factor`*>

Specifies the factor by which the capacitance of the combinational signals must be scaled for the specified instance. The data type of the scaling factor is `float`.

-comb_leakage_power <*`power_scaling_factor`*>

Specifies the factor by which the combinational leakage power must be scaled for the specified instance. The data type of the scaling factor is `float`.

-comb_internal_power <*`power_scaling_factor`*>

Specifies the factor by which the combinational internal power must be scaled for the specified instance. The data type of the scaling factor is `float`.

-comb_switching_power <*`power_scaling_factor`*>

Specifies the factor by which the combinational switching power must be scaled for the specified instance. The data type of the scaling factor is `float`.

-comb_glitch_derating <*`glitch_derating_factor`*>

Specifies the factor by which the combinational glitch activity must be scaled for the specified instance. The data type of the scaling factor is `float`.

-comb_frequency <*freq_scaling_factor*>

Specifies the factor by which the combinational frequency must be scaled for the specified instance. The data type of the scaling factor is `float`.

-comb_cell_sizing_level <*comb_cell_sizing_level*>

Specifies the level by which the combinational elements must be upsized or downsized in the mapped netlist for the specified instance. The data type of the scaling factor is `integer`.

Related Commands

report_power

Usage

Specifies the constraints that can be used to scale the different parameters of power. These parameters are used to compute the design power.

Examples

Example 1: Scales the capacitance of the flop-driven signal by `2.0` for the top hierarchy and by `3.0` for the instance `inst_a`.

```
set_scaling -flop_switching_cap 2.0
set_scaling -flop_switching_cap 3.0 -inst inst_a
```

Example 2: Scales the capacitance of the combinational signal by `3.0`.

```
set_scaling -comb_switching_cap 3.0
```

Example 3: Upsizes the buffers that are present in the top hierarchy by `2` levels and downsizes the buffers that are associated with the hierarchical instance `inst_a` by `1` level in the clock tree synthesis.

```
set_scaling -clocktree_buffer_sizing_level 2
set_scaling -clocktree_buffer_sizing_level -1 -inst inst_a
```

set_scan_enable

Deprecated w.e.f. PowerPro 10.4.

set_scan_enable — Specifies the signal to be connected to the scan pin of the CGIC cell that is gating the memory clock. If the scan signal has been specified using `set_cell_as_memory`, `set_scan_enable` does not override the signal.

Synopsis

```
set_scan_enable  -signal < scan_signal>
    { -clock < ideal_clock> | -default }
```

-signal *<scan_signal>*

Specifies the signal to be connected to the scan pin of CGIC cell that is gating the memory clock.

-clock *<ideal_clock>* | -default

Specify `-clock` to associate the *`<ideal_clock>`* with the scan signal. The `-clock` option overrides all settings applied using the `set_cell_as_memory` command.

Specify `-default` to connect the scan signal `scan_signal` to the scan pin of the CGIC.

Related Commands

set_cell_as_memory, set_module_as_cgic, set_target_cgic

Usage

The **set_scan_enable** command specifies the signal to be connected to the scan pin of the CGIC cell that is gating the memory clock.

Consider the following example.

```
build_design design.v memory.v
build_design  -cons cgic.v
create_clock  -name CLK  -period 123.45 clk
set_module_as_cgic  -module my_CG_MOD  -clock ck_in \
    -clock_gate_enable_pin enable  -clock_gate_out_pin ck_out \
    -clock_gate_test_pin test
set_scan_enable -default  -signal SCAN_EN
set_cell_as_memory  -module p1024x32m \
     -enable_polarity_clock "me pos clk"  -clock_and_polarity "clk pos"
prototype_design
```

In this example,:

- The `set_scan_enable` command specifies that the scan signal `SCAN_EN` be used to connect to the scan pin of the CGIC cell `my_CG_MOD`.
- The instance of the CGIC cell `my_CG_MOD` will be used to perform memory clock gating on the memories of the module `p1024x32m`.

set_static_signal

Deprecated w.e.f. PowerPro 10.4.

set_static_signal — Specifies the names of the signals to be used during Static Signal-Based Gating.

Synopsis

```
set_static_signal <Tcl_list_of_signals>
```

`<Tcl_list_of_signals>`

> Specifies a Tcl list of signals to be used by the `insert_static_signal_gating` command for generating gating expressions during Static Signal-Based Gating. Signals specified with this command are assumed to be static through simulation.

Related Commands

insert_static_signal_gating, report_static_signal_gating, report_enable_expression

Usage

The `set_static_signal` command specifies the names of the signals to be used by the `insert_static_signal_gating` command. A static signal is a signal that does not change value during simulation.

A sample flow is shown below.

```
prototype_design
...
//Specifies the signals to be used for generating gating expressions
//during Static Signal-Based Gating
set_static_signal {sig1 sig2 sig3}

//Generates Gating expressions based on the static signals 'sig1',
'sig2' and 'sig3'
insert_static_signal_gating

//Returns only those expressions generated during Static Signal-Based
//Gating. Same as specifying report_enable_expression  -static
report_enable_expression  -all
...
//Call Sequential Optimization Commands
<sequential_optimization_commands>

//Returns gating expressions generated during Static Signal-Based
//Gating as well as domains committed during sequential optimization
report_enable_expression  -all
```

Example

Example 1: Specifies that the signals `sig1`, `sig2` and `sig3` be used by `insert_static_signal_gating` for determining potential static signal-based gating opportunities.

```
set_static_signal {sig1, sig2, sig3}
insert_static_signal_gating
report_enable_expression  -static
...
```

set_target_cgic

Deprecated w.e.f. PowerPro 10.4.

set_target_cgic — Specifies the CGICs to be used for clock gating memory instances.

Synopsis

```
set_target_cgic <cgic_name> [ -clock<ideal_clock> |  -default ]
    {  -pos |  -neg } [ -mem_inst<memory_instance>]
```

<cgic_name>

Specifies the clock gating module to be used for gating the clock ports of the target memories. The module can be a CGIC cell that has already been defined using the `set_module_as_cgic` command or a CGIC cell whose definition is present in the library being read.

-clock*<ideal_clock>* | -default

Use `-clock` to associate the CGIC cell with the *<ideal_clock>* specified.

Use `-default`, to use the CGIC cell *<cgic_name>* as the default.

Note: In case the `-clock` option is specified after the `-default` option was specified, `-clock` will take precedence. The `-clock` option also overrides any settings applied using the `set_cell_as_memory` command.

-pos | -neg

Indicates the polarity of the clock to be gated. *This option requires the* *-clock* *option to be specified.*

Specify `-pos` to gate all user memory clocks driven by positive phase of *<ideal_clock>*.

Specify `-neg` to gate all user memory clocks driven by negative phase of *<ideal_clock>*.

If this option is not provided, `set_target_cgic` can gate both positive and negative phase driven memory clocks.

-mem_inst*<memory_instance>*

Specifies the CGIC to be used for gating memory instances of memory modules which have already been specified using the `set_cell_as_memory` command.

Related Commands

insert_mem_clock_gating, report_clock_gated_memories, set_cell_as_memory, set_module_as_cgic, set_scan_enable

Usage

The `set_target_cgic` command is used to explicitly specifies the CGICs to be used for clock gating memory instances.

Consider the following example. This example shows how the `set_target_cgic` command is used in a memory clock gating flow, as well as the order of the commands.

```
build_design design.v memory.v
build_design -cons cgic.v
create_clock -name CLK  -period 123.45 clk
prototype_design

set_module_as_cgic -module my_CG_MOD  -clock ck_in
-clock_gate_enable_pin enable  -clock_gate_out_pin ck_out
-clock_gate_test_pin test
set_module_as_cgic  -module my_CG_MOD2  -clock ck_in2
-clock_gate_enable_pin enable2  -clock_gate_out_pin ck_out
-clock_gate_test_pin test

set_scan_enable -default  -signal SCAN_EN

set_cell_as_memory  -module p1024x32m  -enable_polarity_clock "me pos
clk"  -clock_and_polarity "clk pos"

set_target_cgic -default my_CG_MOD

set_target_cgic -mem_inst p1024x32m_inst2 my_CG_MOD2

insert_mem_clock_gating
```

In this example,

- The CGIC cells `my_CG_MOD` and `my_CG_MOD2` are being used for clock gating, where `my_CG_MOD` will be used as the default CGIC to perform memory clock gating.
- The CGIC cell `my_CG_MOD2` will be used to clock gate the memory `p1024x32m_inst2`.
- The `set_cell_as_memory` command has been used to specify the `p1024x32m` module as the power-opt memory module to be clock gated.

set_target_technology

set_target_technology — Specifies the libraries to be used for mapping parts of the design. Note that the technology libraries must have been read using the `read_library` command prior to calling the `set_target_technology` command. Additionally, this command can only be called before `prototype_design`.

Synopsis

```
set_target_technology -library <library_list>
    [ -inst <instance_list>] [ -list]
```

-library *<library_list>*

Specifies the libraries to be used for mapping the instances specified by `-inst`. These libraries must have been read using the `read_library` command.

-inst *<instance_list>*

Specifies a list of hierarchical names for instances on which the constraint is to be applied. If this option is not specified, the constraint is applied to all design objects.

-list

Displays a list of technology libraries that have been read. When this option is specified, all other options are ignored.

Related Commands

report_power, set_binding, set_operating_condition

Usage

The `set_target_technology` command allows one to specify different technology libraries for mapping parts of the design.

Note that the `set_target_technology` command implicitly sets the operating conditions for the specified instance to the default operating condition of the first library specified with the command. For example, specifying the command `set_target_technology -library "lib2 lib3" -inst f1.f2`, will apply the default operating condition in the library `lib2` to the instance `f1.f2`.

To get a list of all target technology libraries read, use the `-list` option. Note that when this option is specified, the `set_target_technology` command only displays a list of technology libraries and does not apply any constraints.

If the target technology libraries have not been specified for a hierarchical instance, it is derived from the parent hierarchy. Consider the following design.

```
//File: file1.lib

library(lib1)
{
...

//File: file2.lib

library(lib2)
{
...

//File: file3.lib

library(lib3)
{
...

//File: top.v
module inner_level1
    inner_level2 f2(...);
    ...
endmodule

module top
    inner_level1 f1 ( ..);
    ...
endmodule
```

In this example, the design objects in `f1.f2` will be mapped using the library `lib3`; the remaining design objects within `f1` will be mapped using the library `lib2`. For all other remaining design objects, the library `lib1` will be used for mapping.

```
set_target_technology -library lib2 -inst f1
set_target_technology -library lib3 -inst f1.f2
set_target_technology -library lib1
```

Examples

Example 1: Maps the instance `f1.f2` using the technology library `lib3`. All other design objects are mapped using all the libraries been read using the `read_library` command. In this case, the rest of the instances are mapped using `lib1` and `lib3`.

```
read_library file1.lib  //contains lib1
read_library file3.lib  //contains lib3
...
set_target_technology -library lib3 -inst f1.f2
...
prototype_design
```

Example 2: Maps the instances `f1.f2` and `f3` using the technology libraries `lib2` and `lib3`. All other instances in the design are mapped using the libraries `lib1`, `lib2` and `lib3`. Additionally, `set_target_technology` implicitly applies the default operating condition of the library `lib2` (first library specified with `set_target_technology`) on the instances `f1.f2` and `f3`. For all other instances, the operating condition `oc1` from `lib1` is applied.

```
read_library file1.lib  //contains lib1
read_library file2.lib  //contains lib2
read_library file3.lib  //contains lib3
...
set_operating_condition -library lib1 oc1
...
set_target_technology -library "lib2 lib3" -inst "f1.f2 f3"
...
prototype_design
```

set_threshold_voltage_group

set_threshold_voltage_group — Sets the `threshold_voltage_group` attribute on the specified cells. Note that this command can only be called after `read_library` and before `prototype_design`.

Synopsis

```
set_threshold_voltage_group -name <group_name>
    -cells <list_of_cells> ...
```

-name *<group_name>*

Specifies the name of the threshold voltage group to set as the `threshold_voltage_group` attribute on the specified cells.

-cells *<list_of_cells>*

Specifies the cells on which the `threshold_voltage_group` attribute is to be set. Name should be specified in directory structure format i.e., the cell name should be preceded by library name and separated by a / . For example, `lib1/AND1`.

Note: The `find_lib_cell` command can be used to get a list of cell names.

Related Commands

find_lib_cell, set_default_threshold_voltage_group, set_multi_vth_constraint

Usage

The `set_threshold_voltage_group` command sets the `threshold_voltage_group` on given cells.

Examples

Example 1: Sets `hvth` as the `threshold_voltage_group` attribute on the cells `AND1` and `OR1` in the library `lib1`

```
set_threshold_voltage_group -name "hvth" -cells "lib1/AND1 lib1/OR1"
```

Example 2: Sets `lvth` as the `threshold_voltage_group` attribute on all cells whose name starts with `ANDLVT` in the library `lib1`.

```
set_threshold_voltage_group -name "lvth"  -cells [find_lib_cell
lib1/ANDLVT*]
```

set_transition

set_transition — Sets the specified slew (max) values on the given ports. Note that this command can only be called before `prototype_design`.

Synopsis

```
set_transition -rise<time in ns> -fall<time in ns> <list of ports>
```

-rise<*time in ns*>

Specifies rise slew (max) in ns.

-fall<*time in ns*>

Specifies fall slew (max) in ns.

<*list of ports*>

Specifies the list of port names separated by whitespaces.

Related Commands

get_transition

Usage

Helps in setting the slew values for the ports.

Examples

Example 1: Sets the slew for the port abc.in with the rise slew value as 10 ns and the fall slew value as 5 ns.

```
set_transition -rise 10 -fall 5 abc.in
```

set_unobservable

set_unobservable — Sets the specified user port or signal as unobservable. This information is used during observability-based sequential optimization. Note that this command can only be called after the `prototype_design` command.

Synopsis

```
set_unobservable { -module <module_name> |  -instance }
    {-port <path_to_user_port> | -signal <path_to_user_signal> }
```

-module *<module_name>* | -instance

Specify the `-module` option to apply the constraint to the specified port or signal in all instances of the module *<module_name>*.

Specify the `-instance` option to apply the constraint to a specific port or signal within the instance. When specifying this option, ensure that the port name specified by the `-port` option or the signal name specified by the -signal option is hierarchical.

-port *<path_to_user_port>* | -signal *<path_to_user_signal>*

Specifies the name of the user port or signal on which the constraint is to be applied.

Note: In case the `-instance` option has been used, specify the hierarchical path of the port or signal.

Related Commands

insert_observability_logic, set_observable

Usage

The **set_unobservable** command sets a design port as unobservable. This information is used by observability based sequential optimization while computing CG- or MG-based moves.

Examples

Example 1: Sets the port `row` as unobservable in all the instances of module `two_tap`.

```
set_unobservable -module two_tap  -port row
```

Example 2: Sets the port `start` of the instance `new_inst` as unobservable.

```
set_unobservable  -instance  -port new_inst.start
```

set_user_synchronizer

set_user_synchronizer — Replaces the default Calypto synchronizer with a user synchronizer. This command can only be called after `prototype_design` and before any sequential optimization command.

Synopsis

```
set_user_synchronizer -module <module_name> { -data | -reset }
    { -pos | -neg } -clock <clkPort> -resetin <resetInPort>
    -out <resetOutPort> [-others <portMapStr> ]
```

-module *`<module_name>`*

Specifies the name of the user synchronizer module.

Note that the design containing the module must be read-in using the `build_design -cons` command before issuing the `set_user_synchronizer` command.

-data | -reset

Specify `-data` to indicate that both edges are synchronized.

Specify `-reset` to indicate that only the deassert is synchronized.

-pos | -neg

Specifies the polarity of the synchronizer module.

Specify `-pos` for active high signals and `-neg` for active low signals.

-clock *`<clkPort>`*

Specifies the clock port of the module.

-resetin *<resetInPort>*

Specifies the name of the input reset port of the module.

-out *<resetOutPort>*

Specifies the name of the output reset port of the module.

-others *<portMapStr>*

Specify `-others` to map other input ports of the module to their corresponding design ports. The first port in the mapping refers to synchronizer instance port while the second port refers to the design port.

Related Commands

build_design, prototype_design

Usage

The `set_user_synchronizer` command replaces the default Calypto synchronizer with the user synchronizer.

Examples

Example 1: Creates a positive reset synchronizer module `sync_mod`. The input clock port of the module is `clk`. The input reset port is `rstin` and the output reset port is `rstout`.

```
build_design -cons sync_mod.v
....
prototype_design
set_user_synchronizer -module sync_mod -reset -pos -clock clk
      -resetin rstin -out rstout
....
```

Example 2: Creates a negative data synchronizer module `sync_mod`. The input clock port of the module is `clk`. The input reset port is `rstin` and the output reset port is `rstout`. The extra port `in1` is connected to the `in1` port of the design while `in2` is connected to the `ins1.in2` port of the design.

```
build_design -cons sync_mod.v
....
prototype_design
set_user_synchronizer -module sync_mod -data -neg -clock clk
     -resetin rstin -out rstout -others {{in1 in1} {in2 ins1.in2}}
....
```

set_verbosity

set_verbosity — Sets the amount of information reported by PowerPro for each message.

Synopsis

```
set_verbosity { -error <message_id> | <message_id> <level> | <level> }
```

-error *<message_id>*

Upgrades the severity of a Warning to an Error.

<message_id> <level>

A value between 0 and 9. This value overrides the default verbosity level of a message and is used to determine whether or not a message will be displayed.

<level>

A value between 0 and 9 determining whether or not a message will be displayed. For example, if an error message has a verbosity level of 4 and this value is set to 3, the message will not be displayed. On the other hand, if this value is set to 6, this message will be displayed.

Usage

This command controls the amount of information reported by PowerPro.

Every message in PowerPro has a default severity level. By specifying the `-error` option, a warning message can be upgraded to an error message. For example, to upgrade the severity of the message ELEC-EWU to an error, specify the following command:

```
set_verbosity  -error ELEC -EWU
```

Each message in PowerPro has a default verbosity level and is displayed only if the verbosity level set using the `set_verbosity` *<level>* command is set to that number or lower. For example, this command sets the verbosity level to 5. By specifying this command, all messages with a default verbosity level of 5 and below are displayed.

```
set_verbosity 5
```

The verbosity level of a specific message can also be changed. For example, this command changes the verbosity level of the message CDB-INF to 8. This message will be displayed only if `set_verbosity  -level` is set to 8 or lower.

```
set_verbosity CDB -INF 8
```

set_voltage

set_voltage — Applies operating voltage to a list of modules, instances or supply objects. Note that this command can only be called after `load_upf`.

Synopsis

```
set_voltage <voltage>
    { -module <tcl_list_of_modules> | -inst <tcl_list_of_instances> |
     -object_list <tcl_list_of_supply_objects>}
```

<voltage>

Specifies the operating voltage to be applied to the specified objects.

-module *<tcl_list_of_modules>* | -inst *<tcl_list_of_instances>* | -object_list *<tcl_list_of_supply_objects>*

Specify `-module` to apply the operating voltage to all instances of the specified modules.

Specify `-inst` to apply the operating voltage to a list of instances.

Specify `-object_list` **to apply the operating voltage to a list of supply ports in form of** `supply_port` **and supply nets in form of** `supply_net`, `supply_set.supply_net` **and** `power_domain.supply_set_handle.supply_net`**.**

Related Commands

load_upf

Usage

The **set_voltage** command applies an operating voltage to a list of modules, instances or supply objects (supply nets or supply ports).

Examples

Example 1: Applies a voltage of `.9V` to the supply nets `sn1` and `sn2` and a voltage of `.6V` to the supply port `VDDPI`. Here, it is assumed that the UPF file containing definitions for the supply nets `sn1` and `sn2` and the supply port `VDDPI` has been read prior to calling the `set_voltage` command.

```
set_voltage 0.9 -object_list {sn1 sn2}
set_voltage 0.6 -object_list {VDDPI}
```

Example 2: Applies a voltage of `2.3V` to the supply net of primary supply set handle of `PD_TOP` and a voltage of `2.1V` to the supply net of supply set `ss_top`. Here, it is assumed that the UPF file containing definitions for the power domain `PD_TOP` and the supply set `ss_top` has been read prior to calling the `set_voltage` command.

```
set_voltage 2.3 -object_list {PD_TOP.primary.power}
set_voltage 2.1 -object_list {ss_top.power}
```

Example 3: Applies a voltage of `1.4V` to the hierarchical instances `i0` and `mc`. The voltage will be applied to all instances within `i0` and `mc`. The rest of the design will continue to work on the original voltage.

```
set_voltage 1.4 -inst {i0 mc0}
```

set_wire_load_mode

set_wire_load_mode — Specifies the wire load mode to be used for hierarchies in design. Note that this command can only be called after `build_design` and before `prototype_design`.

Synopsis

`set_wire_load_mode` *`<mode_name>`*

`<mode_name>`

Sets the wire load mode for the hierarchical wire load models. Legal values are: `top` and `enclosed`. Default: top

Specify `top` to calculate the wire capacitance of all nets based on the wire load model set on for the top-level design.

Specify `enclosed` to calculate the wire capacitance of all nets based on the wire load model set on the smallest sub-design that completely encloses that net.

Related Commands

report_power, set_wire_load_model

Usage

The `set_wire_load_mode` command specifies the wire load mode to be used for hierarchies in design. In case the wire load mode is not specified, the mode is set to `top`.

Example

Example 1: Calculates the wire capacitance of all nets based on the wire load model set on the smallest sub-design that completely encloses that net.

```
set_wire_load_mode enclosed
```

set_wire_load_model

set_wire_load_model — Specifies the wire load model to be used for hierarchies in design. Note that this command can only be called after `build_design` and before `prototype_design`.

Synopsis

```
set_wire_load_model <wire_load> [-inst_list <inst_list> ]
    [-library <library_name> ]
```

<wire_load>

Specifies the name of the wire load model to be used. Note that this name must have been defined in the libraries read.

-inst_list *<inst_list>*

Specifies the list of instances the model is to be applied to. If *<inst_list>* is not provided, the model is applied to the top hierarchy.

-library *<library_name>*

Specifies the name of the library containing the wire load model definition.

Related Commands

report_power, set_wire_load_mode

Usage

In case a wire load model is not specified, the value is derived from the wire load selection table defined in the library.

The value used depends on the area of the hierarchy. The hierarchy selected is dependent on the wire load mode specified. In case the mode is set to `top`, the area of the top hierarchy is calculated. In case, the mode is set to `enclosed`, the area of the enclosing hierarchy is calculated.

In case a wire load selection table is not available in the library, the default resistance and capacitance values are applied to the hierarchy.

Example

Example 1: Consider the following library definition.

```
// ***************************************
// file file1.lib
// ***************************************
library (lib1) {
...
wire_load(wl1) {
      resistance  : 0.0;
      capacitance : 0.0001594 ;
      area : 0.0;
      slope : 0.5;
      fanout_length(1, 176.2) ;
      fanout_length(2, 243.6) ;
      fanout_length(3, 329.4) ;
}
wire_load(wl2) {
      resistance  : 0.0;
      capacitance : 0.0002594 ;

      area : 0.0;
      slope : 0.5;
      fanout_length(1, 276.2) ;
      fanout_length(2, 343.6) ;
      fanout_length(3, 429.4) ;
}
```

In this example, specifying the following command will instruct PowerPro to use the wire load model `wl1` for calculating resistance and capacitance values.

```
read_library file1.lib
...
set_wire_load_model wl1 -library lib1
...
```

Example 2: Sets the wire load model to `wl2` for the instance `sub1`.

```
set_wire_load_model wl2 -library lib1 -inst sub1
```

show_analyzer

show_analyzer — Launches PowerPro Graphical User Interface (GUI), which can be used to view and analyze the results generated by PowerPro. The GUI can also be used to view the schematic, reports, and waveform redundancies.

Synopsis

show_analyzer [-name *<window_name>*] [-dashboard *<dashboard_name>*] [-dir *<golden_work_dir>*] [-designfile *<design_file>*] [-qwave *<db_file>*] [-restore *<work_dir>*] [-stage *<stage>*]

-name *`<window_name>`*

Specifies the name that must be appended to the default name of the GUI window. The default name is `PowerPro`, followed by a count. Therefore, the name of the first window that opens is `PowerPro-(1)`. The subsequent windows that open are `PowerPro-(2)`, `PowerPro-(3)`, and so on.

If the `-name` option is specified, the value of the <*`window_name`*> argument must be appended to the existing name. For example, if you run the command `show_analyzer -name PROTO`, a window by the name `PowerPro-PROTO(1)` opens. If the command `show_analyzer -name OBS` is run again, the next window will open as `PowerPro-OBS(2)`.

-dashboard <*dashboard_name*>

Specifies the name of the dashboard for which reports are to be generated and dashboard GUI will be loaded. Few available dashboards are DEFAULT and PADEBUG.

-dir <*golden_work_dir*>

Specifies the path of the golden work directory against which PADIFF dashboard will be created. This option can only be used with PADIFF dashboard.

-designfile <*design_file*>

Loads the specified design file. This option can be used to override the design file that was created during the PowerPro flow.

-qwave <*db_file*>

Loads the specified simulation database file. This option can be used to override the database file that was created during the PowerPro flow.

-restore <*work_dir*>

Specifies the path of the work directory that can be used to restore the information for an old PowerPro run. This option can only be used before `build_design` has run.

-stage <*stage*>

Loads the patched RTL till the specified stage. The acceptable values are `obs`, `mem_obs`, `stb_c`, `stb_s`, `mem_stb_c`, `mem_stb_s`, and `mem_ls`.

Related Commands

write_db

Usage

The `show_analyzer` command launches the PowerPro GUI, which can be used to view and analyze the results generated by PowerPro. The type of reports displayed in the GUI depend on when the `show_analyzer` command is run in the flow.

If the `show_analyzer` command is run after the `build_design` command, the GUI can be used to view the structure of the netlist using the schematic.

If the `show_analyzer` command is run after the `prototype_design` command, then, in addition to the schematic, reports such as User Enabled Flops, Efficiency, Clock Information, Hierarchy Design Metrics, and Area are also displayed.

If the `show_analyzer` command is run after a sequential optimization command, then, in addition to the reports that are displayed when the `show_analyzer` command is run after the `prototype_design` command, the reports showing gating opportunities identified by PowerPro are also available. Additionally, the PowerPro-optimized RTL is displayed only in Classic GUI.

Examples

Example 1: Shows the current status of the PowerPro session. The log files are created in the directory `Analyzer1`. The GUI window is also named Analyzer1.

```
show_analyzer
```

Example 2: Shows the current status of the PowerPro session and creates the log files in the `pre_opt` directory. The GUI window that opens is named `Analyzer1-pre_opt`.

```
show_analyzer -name pre_opt
```

Example 3: Launches the PowerPro GUI using the specified designfile `design.bin`.

```
show_analyzer -designfile /.../design.bin
```

Example 4: Launches the PowerPro GUI using the specified qwave db file `powerleaks.db`.

```
show_analyzer -qwave /.../powerleaks.db
```

Example 5: Launches the PowerPro GUI using data from the work directory `../old_run/calypto`, which was created in a previous PowerPro run.

```
show_analyzer -restore ../old_run/calypto
```

Example 6: Launches the PowerPro GUI for the specified stage `mem_ls`.

```
show_analyzer -stage mem_ls
```

Example 7: Launches the PowerPro GUI with `PADEBUG` dashboard.

```
show_analyzer -dashboard PADEBUG
```

Example 8: Launches the PowerPro GUI with `PADIFF` dashboard to debug against another PowerPro run `golden_run/calypto`.

```
show_analyzer -dashboard PADIFF -dir golden_run/calypto
```

Example 9: Launches the PowerPro GUI with `PADIFF` dashboard using data from the work directory `old_run/calypto` to debug against another PowerPro run `golden_run/calypto`.

```
show_analyzer -restore old_run/calypto -dashboard PADIFF -dir
golden_run/calypto
```

source

source — Reads commands from the specified file and executes them in the Tcl shell.

Synopsis

```
source [ -echo |  -quiet |  -verbose ] <filename>
```

-echo

Displays the sourced content while executing the commands. The content is displayed on the screen as comments and is also written to `powerpro.log` and the `workdir/powerpro.cmd` file as comments.

-quiet

Suppresses any and all recording of the sourced content to the log files.

-verbose

Same as -echo.

<filename>

Specifies the name of the Tcl file containing the commands to execute.

Usage

The **source** command reads commands from the specified file and executes them in the Tcl shell. It is an enhanced version of the standard Tcl `source` command. If none of the optional arguments are specified, the contents of the sourced file are not echoed to the screen or the log file, but they are written to the `powerpro.cmd` file as comments.

Examples

Example 1: Executes all commands in the file `create_bbox.tcl`.

```
source create_bbox.tcl
```

Example 2: Executes all commands in the file `create_bbox.tcl` without leaving traces in the log files.

```
source  -quiet create_bbox.tcl
```

update_read_design

update_read_design — Reads -in the design files required by `update_rtl`.

Synopsis

```
update_read_design [ -verilog |  -vhdl |  -sv ] <arguments>...
```

-verilog | -vhdl | -sv

Design language of the files being read in. If omitted, the suffix of the first file read in is used to establish the design language (Verilog if `.v`, VHDL if `.vhd*` and SystemVerilog if `.sv`).

Verilog and SystemVerilog Arguments

+define+*<macro>* [=*<macro_body>*]

Define a macro. For example, to assign the value `1024` to the macro `WIDTH` when the file `input.v` is read in, specify the following command:

```
update_read_design +define+WIDTH=1024 input.v
```

-f*<list_file>*

Use a list file which contains a set of files and options to be read in.

-y*<dir_name>*

Search for unresolved modules in the directory *<dir_name>*.

-v*<file_name>*

Search for unresolved modules in the file *<file_name>*.

+libext+*<ext>*

Search for unresolved modules in a library directory using extension *<ext>*.

+librescan

Search from the beginning of the library list for undefined modules, and force multiple searches in the same library.

+incdir+*<dir>* ...

Search + separated directories to resolve `include directives.

<filenames> ...

A whitespace separated list of design files to be read into PowerPro which may include absolute or relative directory paths. By default, any additional file references included within the design files are searched for in the current working directory, and additional include paths may be included using `+incdir+<dir>` options.

VHDL Arguments

-library *<library>*

Specify that the VHDL units being read in belong to the VHDL logical library *<library>*.

Related Commands

update_rtl

Usage

The `update_read_design` command reads in a set of design files required by the `update_rtl` command.

Examples

Example 1: The following command shows how to read in a set of design files for use with `update_rtl`.

```
update_read_design -verilog dir1/dir2/input1.v  dir1/input2.v
```

Example 2: This example shows how to read in a SystemVerilog file if the design file has a `.v` extension.

```
update_read_design -sv  design.v
```

update_rtl

update_rtl — Associates the optimized or original RTL from the previous session.

Synopsis

```
update_rtl [ -script <path_to_script>] [ -wrtl_dir <path>] [ -original]
```

-script *<path_to_script>*

Changes to the path of the RTL, after the database file has been generated mandates that the new path be reassociated with the database file to ensure consistency of the database file. The `update_rtl` command uses a Tcl script to manage changes to the path of the RTL.

PowerPro generates a sample script file, by the name of `orig_build_script.tcl`, containing the arguments for the original association in the *<work_dir>* directory. This file includes arguments for the association of the original RTL file. Modify this script file to reflect the current location of the RTL and use the `-script` option to specify the path to the Tcl script.

-wrtl_dir *<path>*

Specifies the directory which contains the PowerPro optimized RTL consistent with the database file. When this option is specified, `update_rtl` copies over the optimized RTL files from the specified directory to the `$workdir/calypto_insert_`*<opt>* directory, where *<opt>* is the type of optimization previously performed. In case this option is not specified, `update_rtl` regenerates the optimized patched RTL files in the `$workdir/calypto_insert_`*<opt>* directory.

Note that the `-wrtl_dir` option is ignored when the `-original` option is specified.

-original

Indicates that the path provided to the `update_rtl` command corresponds to the current path of the original RTL (without powerpro optimizations) related to the session in which the database passed to `read_eco_db` was generated.

Related Commands

read_db, read_eco_db, update_read_design, write_rtl

Usage

The `update_rtl` command is used to associate the optimized RTL from the previous session, if sequential optimization had been performed before writing the database file. In case the path to the original RTL changes after the database file was generated, `update_rtl` is used to reassociate the new path of the RTL to the database file.

Note: This command can only be issued if `read_db` has been performed in the current session. In case sequential optimization has been performed in the `write_db` flow, the optimized RTL from the previous session must be reassociated using this command.

Examples

Example 1: Running `update_rtl` without any arguments, reassociates the original RTL. In case of an optimized RTL, the PowerPro optimized patched RTL files are regenerated in the $workdir/calypto_insert_*<opt>* directory, where *<opt>* is the type of optimization previously performed.

```
update_rtl
```

Example 2: Reassociates the path specified in `new_path.tcl` with the database file.

```
update_rtl  -script new_path.tcl
```

Example 3: Reassociates all objects in the database file to the current path of the RTL. It also regenerates the PowerPro optimized patched RTL files in the `$workdir/calypto_insert_<opt>` directory, where *<opt>* is the type of optimization performed previously. A check is also performed to make sure that the new design files are similar in contents to the original RTL; in case of a difference an error is issued.

```
update_rtl  -wrtl_dir obs_run/powerpro_rtl
```

Example 4: Reassociates the path specified in `build_script.tcl` with the database file read during `read_eco_db`.

```
read_eco_db orig.db
update_rtl  -original  -script build_script.tcl
```

---

validate_streaming_data

validate_streaming_data — Validates the reconstructed waveform data streamed by Veloce. The command must be run after the `report_power` command. The output of this command is stored in the `<work_dir>/veloce_post_validation.log` file.

Synopsis

```
validate_streaming_data [-filename <file_name>] [-
    session_tag <session_tag>]
```

-filename <*file_name*>

Specifies the name of the STW file. If the STW file is not specified, PowerPro automatically infers it from the output of the `read_veloce_dataset` command.

-session_tag <*session_tag*>

Specifies the Veloce session tag generated during the `report_power` command run. If the session tag is not specified, PowerPro automatically infers it from the output of the `report_power` command.

Related Commands

validate_trace_for_ppro

Usage

The `validate_streaming_data` command validates the reconstructed waveform data and reports the errors in case there is an issue in the STW streaming process. This command must be run after the `report_power` command.

Example

Example 1: Validates the reconstructed waveform data streamed by Veloce.

```
validate_streaming_data
    [PPRO-RVDI] VALIDATION SUCCESSFULL!!
```

validate_trace_for_ppro

validate_trace_for_ppro — Validates that the STW trace directory is good for the PowerPro run. The output of this command is stored in the `<work_dir>/veloce_pre_validation.log` file.

Synopsis

```
validate_trace_for_ppro -filename <file_name> -
    instance_name <instance_name>
```

- filename <*`file_name`*>

Specifies the name of the STW file.

- instance_name <*`instance_name`*>

Specifies the hierarchical name of the top module of the design, as specified in the STW file. The hierarchical name must use the correct hierarchy delimiter so that the name can be found in the STW file. For VHDL and Mixed language designs, specify the hierarchy delimiter '/'.

Related Commands

validate_streaming_data

Usage

The `validate_trace_for_ppro` command ensures that the trace directory provided by the emulation team has been generated with the correct settings to allow a power analysis run in PowerPro.

Example

Example 1: Validates that the STW file `mytrace.stw` is good for Powerpro run. Note that in the following example, the instance name `testbench` is the name of the testbench module, and `dut` is the name of the instance corresponding to the top module of the design within the testbench.

```
validate_trace_for_ppro -filename .../../mytrace.stw -instance_name
testbench.dut
    [PPRO-RVDI] VALIDATION SUCCESSFULL!!
```

visualize_clock_tree

visualize_clock_tree — Prints information about the CGIC cells present in the design. For each CGIC cell, the report is printed in the following format: ICG-Name, Clock-Source-Name, Depth, RICGE (%), AICGE (%), WICGC, Power Reduction (uW), and N(reg_ICG).

These terms are explained below in more detail.

1. ICG-Name: Name of the CGIC cell.
2. Clock-Source-Name: Name of the root clock port.
3. Depth: Depth of the CGIC cell from the root clock port.
4. RICGE (%): Relative Integrated Clock Gating Efficiency (in percentage).

   (N(CK) - N(ECK)) / N(CK).
5. AICGE (%): Absolute Integrated Clock Gating Efficiency (in percentage).

   (N(CK) - N(ECK)) / N(CLK).
6. WICGC: Weight Integrated Clock Gate Impact On Clock Cycle Reduction.

   (N(CK) - N(ECK)) * N(reg_ICG)
7. Power Reduction (uW): Power saved by the CGIC cell.
8. N(reg_ICG)

where,

N(CK) : Number of clock cycles at the input clock port of the CGIC cell.

N(ECK) : Number of clock cycles at the output clock port of the CGIC cell.

N(CLK) : Number of clock cycles at the clock root of the CGIC cell (the clock root is printed in the report).

N(reg_ICG) : Sum of bit numbers of all the registers driven by the CGIC cell.

Note : The information will be printed for only those CGIC cells that have the user SA data available.

Synopsis

```
visualize_clock_tree [-only_zero_td] [-exclude_clk_source_cgics
    <source_clock_roots_to_exclude_list>] [-filename <filename>] [ -
    text | -xml | -csv ] [-csv_delimiter <csv_delimiter>] [-no_wrap]
```

-only_zero_td

Prints information about all the CGIC cells that have zero (0) toggles at the input clock port.

-exclude_clk_source_cgics <*source_clock_roots_to_exclude_list*>

Specifies the list of clock roots, whose associated CGICs must not be printed. To print information about CGICs associated with all the clock roots, use the `report_clocks` command.

-filename <*`filename`*>

Specifies the name of the file in which the information must be printed. If the file exists, the contents of the file are overwritten. If this option is not specified, the information is displayed on the standard output.

-text | -xml | -csv

Specifies the format in which the information will be printed. The information is printed in table format, with a choice of saving the information report in text, XML, or CSV format. By default, the report is printed as a plain text file.

Note: For CSV files, semicolon (;) is used as the default delimiter. To specify a different delimiter, use the `-csv_delimiter` option.

-csv_delimiter <*`csv_delimiter`*>

Specifies the delimiter for the CSV report. By default, semicolon (;) is used as the delimiter.

-no_wrap

The information is printed in fixed-width columns, by default. If the information for a column exceeds the width of the column, it continues to get printed in the next line under the same column. This option prevents the information in columns from getting split across rows, thereby making it easier to view and extract information from the report.

Related Commands

report_clocks

Usage

The **visualize_clock_tree** command prints information about all the CGIC cells present in the design. For each CGIC cell, the information about RICGE, AICGE, WICGC, source clock port, depth from source clock port, and power saving is printed.

Note: The information printed using this command can be used to evaluate the effectiveness of CGIC cells in the clock-tree structure.

Examples

Example 1: Prints in the `clock_tree_report.csv` file the information in csv format with delimiter '`,`' about all the CGIC cells that are present in the design.

```
visualize_clock_tree -filename clock_tree_report.csv -csv -
csv_delimiter ,
```

Example 2: Prints in the `clock_tree_visualization.txt` file the information about all the CGIC cells that are present in the design, excluding the ones that have the clock root as `clk1` or `clk2`.

```
visualize_clock_tree -exclude {clk1 clk2} -filename
clock_tree_visualization.txt
```

Example 3: Prints information about all those CGIC cells that have zero (0) toggles at the input clock port, excluding the ones that have the clock root as `clk1` or `clk2`.

```
visualize_clock_tree -exclude {clk1 clk2} -only_zero_td
```

weaken_enable_logic

weaken_enable_logic — Reports statistics for power, area and efficiency after weakening the enable associated with the observability-based sequential optimization expression. Note that this command can only be called after `insert_observability_logic` if the `generate_guidance` command has been run in the flow.

Synopsis

```
weaken_enable_logic  -enable <enable_port> [ -list_term |
    -minimal_expression |  -weaken_term <tcl_list_of_signals> ]
    [ -power_weight] [ -percentage] [ -filename <filename>]
```

-enable *`<enable_port>`*

Specifies the name of the enable port associated with the observability-based sequential optimization expression. Note that *`<enable_port>`* must be one of the ports reported by the `report_enable_expression` command.

-list_term | -minimal_expression | -weaken_term *`<tcl_list_of_signals>`*

Specify `-list_term` to generate a list of user signals participating in the enable expression.

Specify `-minimal_expression` to identify the weakened expression with the smallest power change. *Note that specifying this option might lead to an increase in runtime of the command as all possible combinations are evaluated. Also note that the `-filename` option cannot be specified with this option.*

Specify `-weaken_term` to weaken the list of signals specified by *`<tcl_list_of_signals>`. To get a valid list of user signals that can be specified with this option, use the `-list_term` option.*

-power_weight

Specifies that the generated report include information about the difference between the power of the weakened expression and the original observability-based sequential optimization expression. This option requires the `-list_term` option to be specified.

-percentage

Specifies that the difference between the power of the weakened expression and the original observability-based sequential optimization expression be reported as a percentage as well. This option requires the `-power_weight` option to be specified.

-filename *<filename>*

Specifies the name of the report file to create. If the file exists, contents are overwritten. If this option is not specified, the report is sent to the standard output. Note that this option requires the `-list_term` or `-power_weight` option to be specified.

Related Commands

insert_observability_logic, report_enable_expression

Usage

The **weaken_enable_logic** command weakens the enable associated with the observability-based sequential optimization expression and reports information like power, area and efficiency post weakening of the enable.

Examples

Example 1: Displays all user signals associated with the enable port `inst_cg_obs_ODC_1.data_in_en`.

```
weaken_enable_logic  -enable "inst_cg_obs_ODC_1.data_in_en"  -list_term
```

Example 2: Displays all user signals associated with the enable port `inst_cg_obs_ODC_1.data_in_en`. Additionally, for each signal, information about the difference in the power of the weakened expression and the original observability-based sequential optimization expression is also displayed. The results are written to a file named `power_weight.txt`.

```
weaken_enable_logic  -enable "inst_cg_obs_ODC_1.data_in_en"   \
    -list_term  -power_weight  -filename "power_weight.txt"
```

Example 3: Displays the customized observability expression for the enable port `inst_cg_obs_ODC_1.data_in_en` after weakening the user signal `sel`.

```
weaken_enable_logic  -enable "inst_cg_obs_ODC_1.data_in_en"    \
          -weaken_term sel
```

write_cgic_module

write_cgic_module — Writes the CGIC module templates in the file.

Synopsis

```
write_cgic_module { -name <PowerPro_CGIC_module> }
    { -type <type_of_CGIC_template> } [ -append ] { <filename> }
    { -verilog |  -vhdl }
```

-name *<PowerPro_CGIC_module>*

Specifies the name of the PowerPro CGIC module to be written.

-type *<type_of_CGIC_template>*

Specifies the type of CGIC template to be written. It is a concatenation of up to three strings: first specifies latch or none, second specifies whether it is appropriate for positive or negative edge-triggered registers and third specifies test control logic. Valid values are `latch_posedge_control`, `latch_negedge_control`, `none_posedge_control` and `none_negedge_control`.

-append

If specified, module template is appended to an already existing file. Else, module template is written in the new file.

<filename>

Name of the file in which module template is to be written. Filename should contain proper file extension corresponding to the Language Modes selected (".v" for verilog and ".vhd"/".vhdl" for VHDL)

-verilog | -vhdl

Specifies the language mode to be set.

Specify `-vhdl` to set the language mode to VHDL. Specify `-verilog` to set the language mode to Verilog.

Related Commands

set_cgic_cell

Usage

The **write_cgic_module** command writes the CGIC module templates in the file. These templates are used to model user provided clock gating module in terms of the PowerPro supported CGIC.

Examples

Example 1: Writes the CGIC module template corresponding to the latch based positive edge triggered clock gating module. The module is named `cgic_mod` and the template is written to the new file `cgic.v`.

```
write_cgic_module  -name cgic_mod  -type latch_posedge_control cgic.v
    -verilog
```

Example 2: Writes the CGIC module template corresponding to the latch based negative edge triggered clock gating module. The entity is named `cgic_mod`. The template is appended to the existing file `cgic.vhd`. If the file does not exist, a new file is created.

```
write_cgic_module  -name cgic_mod  -type latch_negedge_control
     -append cgic.vhd  -vhdl
```

write_db

write_db — Writes a database to a file.

Synopsis

```
write_db <file>
```

<file>

File to write design database.

Related Commands

read_db, show_analyzer

Usage

The **write_db** command can be used to write a design database to a file. The database can be written anytime after running the `build_design` command and can be read in a different session.

Examples

Example 1: Writes the `rtl.db` file after running the `build_design` command for `four` verilog files.

```
build_design rtl_main.v rtl2.v rtl3.v rtl4.v
write_db rtl.db
exit
```

write_phy_db

write_phy_db — Writes parasitic information that has been extracted from SPEF to a file.

Synopsis

`write_phy_db <`*`file`*`>`

`<file>`

The file in which the parasitic information must be written.

Related Commands

read_phy_db, read_spef

Usage

The `write_phy_db` command is used to write parasitic information to a file. The command can be used only when the commands `read_spef` and `prototype_design` have been executed. The file can be read in a different session.

Note: This command is under limited production support.

Examples

Example 1: Writes parasitic information to the `rtl_main.phydb` file.

```
build_design rtl_main.v
read_spef rtl_main.spef
prototype_design
write_phy_db rtl_main.phydb
exit
```

write_rtl

write_rtl — This command produces power-optimized RTL files based on original RTL files, by applying the optimizations implemented by `insert_observability_logic` and/or `insert_stability_logic`.

Synopsis

```
write_rtl [ -output_dir <path>] [ -force ]
```

-output_dir *<path>*

Specifies the name of the directory where the output files are to be written.

-force

When the name of the user specified output directory is the same as the name of the directory implicitly generated by the `write_rtl` command after each sequential optimization command, the directory will not be overwritten. To overwrite such directories, use the `-force` option with the `-output_dir` option.

Related Commands

insert_mem_observability_gating, insert_mem_sleep_gating, insert_mem_stability_gating, insert_observability_logic, insert_stability_logic, read_design

Usage

This command can be used to apply the optimizations done by `insert_observability_logic` or `insert_stability_logic` or the combination of both, in the original design after which we get a new RTL patched with the optimizations.

Examples

```
write_rtl -output_dir ppro_rtl
```

The execution of this command will write the patched RTL files in the <pwd>/ppro_rtl directory.

```
write_rtl  -output_dir calypto/calypto_insert_obs  -force
```

As the specified output directory "calypto/calypto_insert_obs" is internally generated through the implicit write_rtl call after insert_observability_logic command, to overwrite it, `-force` switch is used.

write_saif

write_saif — Writes switching activity information from a design as a backward Switching Activity Interchange Format (SAIF) file. Note that this command can only be called after the `prototype_design` command.

Synopsis

```
write_saif { -filename<filename> } [ -all | -user | -tool_computed ]
    [ -all_signal | -user_signal | -rtl ] [ -timescale<timescale>]
    [ -duration<duration>] [ -propagate { on | off } ]
    [ -instance_name <instance> ] [ -name_divider <name_divider> ]
```

-filename*<filename>*

Specifies the name of the output SAIF file.

-all | -user | -tool_computed

Specifies the assertion type of edges for which switching activity information is to be written to the SAIF file. Default is `all`.

-all_signal | -user_signal | -rtl

Specifies the type of signal for which switching activity information is to be written to the SAIF file. Default is `user_signal`.

all_signal

Specifies that switching activity information be written for all signals in the design, including internal signals.

user_signal

Specifies that switching activity information be written only for user signals and CG/MG signals in the design.

rtl

Specifies that switching activity information be written only for synthesis invariant objects. These include design ports, hierarchical cell pins, and outputs of sequential cells.

-timescale *<timescale>*

Specifies the timescale. This option can be specified only if the `duration` option is specified. This option can accept only float values. Default value of this option is `100 ns`.

-duration *<duration>*

This option can be specified only if the `timescale` option is specified. This option can also accept float values. Default value of this option is `100000`.

-propagate { on | off }

This will trigger SA Propagation to estimate switching activity on all the nets, and can be runtime intensive. Default is `on`.

-instance_name *<instance>*

The SAIF file is generated assuming the current design top is instantiated directly in the testbench. If that is not the case, this option can be used to create wrapper INSTANCE scopes in generated SAIF file. Same instance_name value should be used while trying to read back the generated SAIF file.

-name_divider

Specifies the name divider to be used for hierarchical names. This option works only for hierarchical names specified with `-instance_name`. Default is ".".

Related Commands

get_sa, read_saif

Usage

The **write_saif** command writes out switching activity information in SAIF format.

Examples

Example 1: Writes switching activity information to the file `example.saif` for all user signals irrespective of assertion type on edge.

```
write_saif  -filename example.saif
```

Example 2: Writes switching activity information to the file `example.saif` for all signals irrespective of assertion type on edge.

```
write_saif  -filename example.saif  -all_signal
```

Example 3: Writes switching activity information to the file `example.saif` for all signals which has user asserted sa data.

```
write_saif  -filename example.saif  -all_signal  -user
```

write_verify_script

write_verify_script — Generates a Tcl file for SLEC to verify PowerPro optimizations and RTL output. Note that this command can only be called after calling sequential optimization commands.

Synopsis

```
write_verify_script [ -self <filename>] [<output_dir>]
    [ -dont_stop_on_aborted_verification]
```

-self *`<filename>`*

Specifies that PowerPro write an identity -check setup to the file specified by *`<filename>`*. When this option is specified, this command cannot be called before `prototype_design`.

`<output_dir>`

Specifies the name of output directory where the Tcl script and its support files are to be written.

-dont_stop_on_aborted_verification

Specifies that PowerPro generate verification scripts to allow multiple stage verification without aborting even when verification fails for a stage. For failed stages, PowerPro generates the `powerpro_hard_maps_all.tcl` file which can then be sourced in a subsequent PowerPro run to get a full proof.

Related Commands

insert_mem_clock_gating, insert_mem_observability_gating, insert_mem_sleep_gating, insert_mem_stability_gating, insert_observability_logic, insert_stability_logic, write_rtl

Usage

The **write_verify_script** command generates Tcl files which check the modified output of PowerPro after `insert_observability_logic` or `insert_stability_logic`.

Examples

Example 1: Generates Tcl files to verify the output of `write_rtl` against the original design, in the presence of default globals. These Tcl files will be written to the `work_dir/verify` directory.

```
write_verify_script
```

Example 2: Generates a Tcl file by the name of `identity.tcl` for the identity check. This Tcl file will be written to the directory `mydir`.

```
prototype_design
...
write_verify_script  -self identity.tcl mydir
...
```

Chapter 3. Global Variable Reference

alert_color

alert_color — Text color for alert messages such as warnings and errors on color -capable terminals. Legal settings are `black`, `blue`, `green`, `cyan`, `red`, `purple`, `brown`, `gray`, `dark_gray`, `light_blue`, `light_green`, `light_cyan`, `light_red`, `light_purple`, `yellow`, `white` and `off` (to disable the feature). Default setting is `light_red`. Colorization is automatically disabled on non-color-capable interfaces such as output-redirection and output-piping.

Description

Alert messages such as warnings and errors may be displayed in a unique color to highlight their presence. This variable can be set to customize the desired color, depending on the terminal environment.

By default, PowerPro uses `light_red` as the color for warnings and errors sent to color -capable terminals.

analyze_constant_register

analyze_constant_register — Determines whether the constant registers in the design can be considered for power analysis and optimization. Possible values are `0` and `1`. Default value is `0`.

Description

The global `analyze_constant_register` is required to enable power analysis and optimization for constant registers in the design. By default, the power for constant registers is considered to be `0` and gating condition is not generated for these registers. When this global is set to `1`, PowerPro will consider the constant registers as normal registers. Power analysis and optimization then analyzes these registers so that their respective gating condition can be generated resulting in register power saving. The global must be set to `1` if the downstream synthesis flow has been set to preserve the constant registers.

bind_all_to_tech_cell

bind_all_to_tech_cell — Determines how to bind modules that have a technology cell definition and an RTL definition. By default, when both definitions exist, PowerPro automatically binds modules to the RTL definition. Legal values are `0` and `1`. Default is `0`.

Description

The `bind_all_to_tech_cell` global determines how PowerPro binds modules when both the technology cell definition and the RTL definition exist. By default, modules are automatically bound to the RTL definition. To bind modules to the technology cell definition instead of the RTL definition, set this global to `1`.

design_partition_count

design_partition_count — Determines whether to enable the design partition flow in PowerPro. The total number of the partitions would be `1` more than the partition count specified. Possible values are `-1` for partition identification flow, `0` for no partition flow and greater than `0` for design partition flow. Default is `0`.

Description

The `design_partition_count` global determines the number of partitions that must be created to run the design partition flow. By default, the design partition flow is disabled. If the design partition flow is enabled, PowerPro will create one additional partition with the specified number of partitions in the flow. When all the partition runs are complete, the respective reports from different partitions are merged and can be viewed using the `show_analyzer` command.

eco_override_opt_hier_open_signal

eco_override_opt_hier_open_signal — In ECO mode, this global overrides the open signals in opt hierarchy with the global value, if specified with possible values other than –`1`. Possible values are `0`, `1`, `x`, `z`, and `-1`. Default value is `-1`.

Description

The `eco_override_opt_hier_open_signal` global overrides the open signals in opt hierarchy with the global value. Default value is `-1`. Note that there will be no impact on the tool behavior if the default value is set.

---

enable_mixed_language_support

enable_mixed_language_support — Enables mixed language support for reading in Verilog and VHDL designs together. Legal values are `0` and `1`. Default: `1`.

Description

By default, mixed language designs(Verilog and VHDL) are supported in PowerPro. To disable support for reading in mixed language designs, set the `enable_mixed_language_support` global to `0`.

Note: PowerPro does not support SystemVerilog designs in the mixed language flow.

---

enable_powerleak_flow

Deprecated w.e.f. PowerPro 10.6.

enable_powerleak_flow — Determines whether to enable the PowerLeak flow in PowerPro. When set to `1`, the PowerLeak flow will be enabled. Possible values are `0` and `1`. Default value is `0`.

Description

The `enable_powerleak_flow` global determines whether the PowerLeak flow must be enabled in PowerPro. By default, the PowerLeak flow is disabled. Once the Powerleak flow is enabled, the guidance reports will be automatically generated after the `generate_guidance` command is run in the flow. The guidance reports also have the waveform information, which can be viewed using the `show_analyzer` command.

Note: The PowerLeaks flow cannot be called in the vectorless flow or the FSDB scenarios with low assertion.

---

enable_runtime_filter

enable_runtime_filter — Filters flops based on initial efficiency and power, during sequential analysis, to improve PowerPro runtime. Legal values are `0` and `1`. Default:`1`.

Description

By default, PowerPro filters flops based on initial efficiency and power to save runtime. To disable filtering of flops, set this global to `0`.

---

enable_sim_parallel_process

enable_sim_parallel_process — Determines whether the simulation must be run as parallel processes in the flow. Possible values are `0` and `1`. Default is `1`.

Description

This global enables the simulations, which are used to compute the switching activity, to run as parallel processes. This reduces the total run-time of the simulation. When set to `0`, the simulations will run serially in the flow.

---

error_for_missing_modules

error_for_missing_modules - Controls the handling of modules (entities) whose definitions are missing in the design. When set to `0`, tool infers the definition. When set to `1`, if the module is also not specified using the `create_black_box` command, the tool errors out for the missing definition. Possible values are `0` and `1`. Default is `1`.

Description

For a module (entity) whose definition is not present in the design while the usage is present, and the module is not specified using the `create_black_box` command, the tool can either error for the missing definition or infer the definition. This global states whether to error (value=`1`) or generate the definition (value=`0`) for such modules.

---

error_missing_module_definition_in_protected_region

error_missing_module_definition_in_protected_region - The global is used to black-box the modules that have missing definitions in the protected region. Possible values are `0` and `1`. Default is `1`.

Description

The `error_missing_module_definition_in_protected_region` global is used to black-box the modules that have missing definitions in the protected region. When set to `0`, the modules that have missing definitions in the protected region will be black-boxed.

---

exit_on_error

exit_on_error — Forces PowerPro to exit a Tcl script rather than continue debugging in case of premature termination by a script error. Legal values are `0` and `1`. Default is `0`.

Description

By default, PowerPro traps script errors and presents the user with an interactive shell prompt for debugging. When set to `1`, PowerPro terminates on error instead of providing the user with an opportunity to debug.

---

expand_pla_in_gating_expression

expand_pla_in_gating_expression — Determines whether the gating expressions on flops and memories will contain the programmable logic arrays (PLAs). Possible values are `0` and `1`. Default value is `0`.

Description

The global `expand_pla_in_gating_expression` determines whether the gating expressions on flops and memories will contain PLAs. When set to `1`, the gating expressions on flops and memories will not contain any PLAs. When set to `0`, the gating expressions on flops and memories will contain PLAs.

---

host_setup_configuration

host_setup_configuration — The global `host_setup_configuration` can be used only if any processes must be run on the remote machines. Note that to run a process on the remote machine, the network-specific settings must be provided by the various options of the LSF (Load Sharing Facility) command except the script that is required to be run on the remote machine. This script will be provided by PowerPro itself. Possible values include a valid LSF command except the corresponding script, which would be taken care of by PowerPro.

Description

To run a process on the remote machine, the network-specific settings must be provided by the various options of the LSF command and the script that is required to be run on the remote machine will be internally provided by PowerPro.

---

ignore_memory_libcell_binding_check

ignore_memory_libcell_binding_check — Bypasses the compatibility check for a technology cell corresponding to the memory model. Legal values are `0` and `1`. Default is `0`.

Description

When set to `1`, the flow will continue even if there are no technology cells corresponding to the memory model instantiated in the design. However, the power numbers reported for such memory instances would be zero.

---

maximum_iter_count

maximum_iter_count — Sets a limit on the number of iterations to be performed for unrolling immediate loops in a design description. Possible value is any positive integer. Default is `1024`.

Description

An immediate loop is a loop that contains no synchronizing constructs. It executes within one time step. Such loops are unrolled by PowerPro. Setting this global to an integer value establishes a limit on the number of iterations. Exceeding this limit causes an error.

---

max_local_process_usage

max_local_process_usage — Specifies the maximum number of parallel processes that will be used in power computation on the local machine. Possible values are greater than or equal to `zero` (`0`). Default is `0`. In addition to these parallel processes, the main process of PowerPro will also run on the host machine.

Description

This global can be used to configure PowerPro to use multiple cores on the same machine. In PowerPro, Power Analysis and Verification using SLECPro makes use of this feature.

---

max_remote_process_usage

max_remote_process_usage — Specifies the maximum number of parallel processes that will be used in power computation on the remote machine. Possible values are greater than or equal to `zero` (`0`). Default is `0`. In addition to these parallel processes, the main process of PowerPro will also run on the host machine.

Description

This global can be used to configure PowerPro to use multiple cores on a different machine. The setup for LSF or any other workload management system can be customized. The value of the global `host_setup_configuration` must be set correctly to allow the running of jobs on remote machines. In PowerPro, Power Analysis and Verification using SLECPro makes use of this feature.

---

maximum_recurse_count

maximum_recurse_count — Specifies the maximum depth of recursion for function calls in a design description. Legal setting is any positive integer. Default setting is `100`.

Description

Recursive function calls are unfolded and inlined by the PowerPro. By default there is no limit on the depth of recursion. Setting this variable to an integer value establishes a limit on the depth of recursion. If the limit is exceeded then an error message is generated.

---

message_wrap_at

message_wrap_at — Specify the number of characters per line at which the generated messages should be wrapped. Legal settings are any positive integer >= 79. Default setting is `79`.

Description

Use this global to make messages to be formatted based on the size of the window.

---

opt_aol_creation

opt_aol_creation — Controls processing of the asynchronous override logic. Possible values are `0` and `1`. Default is `1`.

Description

The `opt_aol_creation` global controls processing of the asynchronous override logic. To disable processing of the asynchronous override logic, set the global to `0`.

---

opt_call_implicit_write_rtl

opt_call_implicit_write_rtl — Determines whether or not the `write_rtl` command is called during move generation. When disabled, verification scripts are not generated. Legal values are `0` and `1`. Default is `1`.

Description

The `opt_call_implicit_write_rtl` global determines whether or not the `write_rtl` command is called during move generation. By default, PowerPro calls the `write_rtl` command during move generation. To disable the call to `write_rtl`, set this global to `0`.

Note: When this global is disabled, verification cannot be performed and verification scripts will not be generated. If the patched RTL and verification scripts are required, ensure that this global is set to `1`.

---

opt_cg_min_size

opt_cg_min_size — Sets the minimum bit count required for a register or set of registers to be optimized by a PowerPro enable. Default is 4.

Description

Controls the minimum bit-count required for a register or set of registers to be optimized by a PowerPro enable. By default, PowerPro generates an enable only when the bit count of flops gated by it is greater than or equal to `4`.

---

opt_check_dont_opt_at_cdc_boundary_first_receive_flop

opt_check_dont_opt_at_cdc_boundary_first_receive_flop — If this global is enabled, PowerPro marks 'dont optimize' on the CDC boundary flops. If disabled (set to `0`), it errors out when any CDC boundary flops are found. It is disabled by default. This global should be used with global `opt_enable_dont_optimize_cdc_boundary_flop_error`.

Description

PowerPro cannot mark the 'dont optimize' attribute on the clock domain crossings of synchronizer patterns other than CDC. If this global is set to `0`, PowerPro errors out if there are CDC flops without 'dont optimize'. When this global is enabled, it marks 'dont optimize' on the CDC boundary flops.

---

opt_complex_feedback_detection

opt_complex_feedback_detection — Detects complex feedback loop gating expressions for flops during constant stability based sequential optimization. Legal values `0` and `1`. Default is `1`.

Description

The `opt_complex_feedback_detection` global determines whether or not complex feedback loop gating expressions are to be detected for flops during constant stability based sequential optimization.

By default, complex feedback loop gating expressions are detected during constant stability sequential optimization. To exclude such expressions from participating in constant stability based sequential optimization, set this global to `0`.

Note that disabling this global could lead to significant reduction in the number of gating expressions found during constant stability based sequential optimization.

opt_conservative_reset_opt

opt_conservative_reset_opt — Determines whether to remove the flops that have the following load types from the reset-optimization report:

1. Async reset of a flop
2. Synchronizer

Possible values are 0 and 1. Default is 0.

Description

The `opt_conservative_reset_opt` global can be used to remove the flops that have the specified load types from the reset-optimization report. By default, PowerPro does not remove any flops based on their load types from the report.

opt_create_case_insensitive_names

opt_create_case_insensitive_names — Controls the names of PowerPro -generated variables, signals, ports and wires. By default, PowerPro creates unique case insensitive names within the same scope. Legal values are `0` and `1`. Default: `1`.

Description

The `opt_create_case_insensitive_names` global controls the names of PowerPro -generated variables, signals, ports and wires in the original RTL. Setting this global to `0` ensures that PowerPro generated names are case sensitively unique within the scope. Consider the following examples.

Assuming the global `opt_create_case_insensitive_names` is set to `0`, PowerPro creates a wire by the same name but with a different case as shown below.

```
  module ram( ..., ctrl, ... );
  input ctrl;
  wire CTRL; //powerpro added wire
```

When the global `opt_create_case_insensitive_names` is set to `1`, PowerPro generates a unique name for the wire, as shown in the example below.

```
   module ram( ..., ctrl, ... );
   input ctrl;
   wire CTRL_0; //powerpro added wire
```

opt_create_mode_override_with_case_analysis

opt_create_mode_override_with_case_analysis — Controls the creation of mode override logic when the `set_case_analysis` command is run. Possible values are `0` and `1`. Default value is `1`.

Description

This global is used to control the mode override logic when the `set_case_analysis` command is run. If the global is set to `1`, the mode override logic is created during optimization such that whenever any of the constraining signals of the `set_case_analysis` command get a value that differs from its constraining value, all the CG/MG moves are overridden. When set to `0`, the mode override logic is not created when the `set_case_analysis` command is run. However, it will still be created if the user has specified the `create_mode` and `create_mode_constraint` commands directly.

opt_disable_cgic_en2mvm_flop_memory

opt_disable_cgic_en2mvm_flop_memory — Determines whether to perform the cgic enable movement in PowerPro. When set to `0`, the cgic enable movement will be performed. Possible values are `0` and `1`. Default value is `0`.

Description

The `opt_disable_cgic_en2mvm_flop_memory` global determines whether to perform the cgic enable movement in PowerPro. By default, the global is disabled. If the global is enabled, PowerPro will add a black box to the cgic output, and the output of that black box will be considered as the new clock port.

opt_disable_generate_auto_spr

opt_disable_generate_auto_spr — Disables the generation of the new PowerPro reset `POWERPRO_RESET_N` in stability-based optimization (CG/MG), if no explicit `set_powerpro_reset` command is specified. Possible values are `0` and `1`. Default is `0`.

Description

When this global is set to `1`, the `insert_stability_logic/insert_mem_stability_gating` commands do not generate a new PowerPro reset `POWERPRO_RESET_N`, if no explicit `set_powerpro_reset` command is specified. No design reset suggestions as part of `opt_auto_infer_resets.tcl` are generated. The global `opt_skip_pprst_if_async_reset_present` is also disabled.

---

opt_eco_invalidate_same_enable_flops

opt_eco_invalidate_same_enable_flops — Determines how PowerPro handles the flops associated with a PowerPro enable that was invalidated in an ECO run. When set to `1`, PowerPro invalidates all flops associated with that enable. Possible values are `0` and `1`. Default is `1`.

Description

By default, PowerPro invalidates all flops associated with a PowerPro enable that was invalidated in an ECO run. When set to `0`, Powerpro does not invalidate all flops associated with that enable. Only the flop for which the enable is not valid with the ECO changes is invalidated. The directives for the invalid moves, that is, invalidated flops are written to the `eco_directives.tcl` file.

---

opt_dontoptimize_around_memory_flops

opt_dontoptimize_around_memory_flops — When this global is set to 1, all the flops that are connected to the memories till 1 sequential depth are marked with 'dont optimize', and the signals between them are marked with 'dont use attributes'.

Description

Since the interface paths with memory instances can be critical paths, you may not want to optimize the flops connecting with memory instances. In such cases, it is recommended that you enable this global.

---

opt_enable_data_gating

opt_enable_data_gating — Enables support for data gating in the design. Possible values are `0` and `1`. Default is `0`.

Description

The `opt_enable_data_gating` global enables support for data gating in the design. Note that this global can be used only if the `generate_guidance` command has been run in the flow, where it is enabled by default.

---

opt_enable_dont_optimize_cdc_boundary_flop_error

opt_enable_dont_optimize_cdc_boundary_flop_error – PowerPro can recognize only the CDC synchronizer that will not mark 'dont optimize' on the clock domain crossing flops in the design due to some other synchronizer pattern. If this global is enabled, PowerPro checks for other clock domain crossing flops.

Description

When this global is enabled (set to `1`), PowerPro errors out with the list of all the flops at clock domain crossings that are not marked with the 'dont optimize' attribute. Using this global, PowerPro can find the other synchronizer patterns found in the design.

---

opt_enable_frequency_flow

opt_enable_frequency_flow – Enables frequency-based reporting in Clock Gating report. Possible values are `0` and `1`. Default is `0`.

Description

The global `opt_enable_frequency_flow` must be enabled before running the `prototype_design` command. If the global is enabled, the user must resolve all internal clocks in the design using the `set_cgic_cell` or `set_case_analysis` command. The user should explicitly perform `set_frequency` on all user clock roots and CGIC outputs. No vectors can be applied to the design (that is, the usage of `read_fsdb`, `read_saif`, `read_qwave` and `read_vcd` commands is not allowed). Similarly, the global cannot be enabled after applying vectors to the design.

---

opt_ filter_highly_efficient_flops

opt_filter_highly_efficient_flops — Filters flops based on initial efficiency and power to improve PowerPro runtime. Applicable only to Sequential Analysis. Legal values are `0` and `1`. Default is `1`.

Description

By default, during sequential analysis, PowerPro filters flops based on initial efficiency and power to save runtime. If the initial efficiency of the flops is greater than `90%`, PowerPro filters the flops. To disable filtering of flops, set this global to `0`.

---

opt_functional_eco_flow

opt_functional_eco_flow — Determines whether or not the ECO flow is to be enabled. Legal values are `0` and `1`. Default is `1`.

Description

By default, the ECO flow is enabled in PowerPro. To disable the ECO flow, set the `opt_functional_eco_flow` global to `0`.

---

opt_isolate_inout_ports

opt_isolate_inout_ports — Controls support for inout ports in the design. Legal values are `0` and `1`. Default is `1`.

Description

The `opt_isolate_inout_ports` global controls support for inout ports. By default, inout ports are supported in PowerPro. To disable support for inout ports, set this global to `0`.

---

opt_mark_dont_use_enum_edge

opt_mark_dont_use_enum_edge — Determines whether or not edges declared as enums should participate in the creation of the clock gating logic. Legal values are `0` and `1`. Default is `0`.

Description

When the `opt_mark_dont_use_enum_edge` global is enabled, PowerPro applies the dont-use constraint on all edges declared as enum to exclude them from the clock gating logic created by PowerPro.

Consider the following example.

```
//Defines the user enum COLOR.
TYPE COLOR IS ( RED, GREEN );

//Creates a signal 'sel' of type COLOR.
signal sel : COLOR ;
```

In this example, applying the global `opt_mark_dont_use_enum_edge` would cause the signal `sel` to be excluded from the clock gating logic created by PowerPro.

opt_mem_address_coefficient

opt_mem_address_coefficient — This global is used to accurately compute the size of the memory in frequency-based optimization. It is a floating point number with the value ranging from `0` to `1`. Default value is `1`.

Description

The global `opt_mem_address_coefficient` is used as a scaling factor while computing the size of memory in frequency-based optimization. The following equation is used to compute the size of the memory:

```
Memory size = (address-width * data-width) * address-coefficient
```

opt_obs_remove_const_0_moves

opt_obs_remove_const_0_moves — Removes those observability optimizations that have the enable expression as `constant 0`. Possible values are `true` and `false`. Default value is `true`.

Description

The flops that have the enable expression as `constant 0`, that is, the flops that are always disabled functionally are automatically identified and removed by the synthesis tool when some specific flags are set. When this global is set to `false`, such PowerPro optimizations will be available.

---

opt_propagate_static_data_across_flops

opt_propagate_static_data_across_flops — Controls the propagation of static data across flops that do not have feedback loops. Legal values are `0` and `1`. Default is `0`.

Description

The `opt_propagate_static_data_across_flops` global determines whether or not static data, as specified by the `set_static_signal` command, is propagated across flops that do not have feedback loops. By default, the static data property on input pins does not automatically propagate to output pins. To propagate the static property of the input pin to the output pin, set this global to `1`.

---

opt_relax_hard_operator

opt_relax_hard_operator — Controls support for inclusion of large data operators in the gating expression. It is recommended that this global be enabled when the design is operator -intensive. Legal values are `0` and `1`. Default is `0`.

Description

The `opt_relax_hard_operator` global controls how large datapath operators are treated by sequential optimization commands. By default, PowerPro does not include larger datapath operators in the gating expression. In case a large number of signals are reported as violated with the sub -reason `datapath operator`, it is recommended that this global be enabled before calling sequential optimization commands.

---

opt_remove_dead_enable_logic

opt_remove_dead_enable_logic — Determines whether or not PowerPro removes dead clock gating enable logic while running the ECO flow or while incrementally removing clock gating logic. Legal values are `0` and `1`. Default is `0`.

Description

The `opt_remove_dead_enable_logic` global determines whether or not PowerPro removes dead enable logic during the ECO flow or while incrementally removing clock gating logic in the two-pass flow. By default, PowerPro does not remove dead enable logic in these flows.

For more information on the two-pass flow and the ECO flow, refer to the *PowerPro User Manual*.

opt_report_gate_count_change

opt_report_gate_count_change — Determines whether or not information related to change in gate count is to be displayed along with area change information. Legal values are `0` and `1`. Default is `0`.

Description

The `opt_report_gate_count_change` global determines whether or not change in gate count is to be displayed along with the area change. By default, change in gate count is not displayed. To view this information, set this global to `1`.

opt_set_cgic_cell_verification

opt_set_cgic_cell_verification — Determines whether or not PowerPro verifies functional equivalence between the user clock gating module and the PowerPro clock gating module. Legal values are `0` and `1`. Default is `1`.

Description

The `opt_set_cgic_cell_verification` global determines whether or not PowerPro verifies functional equivalence between the user clock gating module and the PowerPro clock gating module. When disabled, the `set_cgic_cell` command does not verify functional equivalence between the user clock gating module and the PowerPro clock gating module.

opt_skip_pprst_if_async_reset_present

opt_skip_pprst_if_async_reset_present — Controls how the `set_powerpro_reset` command treats PowerPro-generated flops that already have an asynchronous reset. Possible values are `0` and `1`. Default: `1`.

Description

When this global is set to `1`, the `set_powerpro_reset` command does not add a reset to PowerPro -generated flops that already have an asynchronous reset. By default, the `set_powerpro_reset` command adds a reset to all PowerPro -generated flops.

---

opt_skip_pprst_if_sync_reset_present

opt_skip_pprst_if_sync_reset_present — Controls how the `set_powerpro_reset` command treats the PowerPro-generated flops that already have a synchronous reset. Possible values are `0` and `1`. Default is `0`.

Description

When this global is set to `1`, no PowerPro reset is added for the PowerPro-generated flops that have a design synchronous reset signal specified as PowerPro reset by the `set_powerpro_reset` command. By default, the `set_powerpro_reset` command adds a reset to all the PowerPro-generated flops. The design synchronous reset signals are marked and identified using the `opt_infer_and_set_sync_reset` command.

---

opt_use_sim_for_clock_gating

opt_use_sim_for_clock_gating — Controls the usage of the PowerPro internal simulation engine to evaluate the clock gating expressions generated by the tool. When set to `0`, PowerPro will not simulate the gating conditions to validate the efficiency improvements. Possible values are `0` and `1`. Default is `1`.

Description

The global `opt_use_sim_for_clock_gating` must be used to evaluate the clock gating expressions generated by the tool using the PowerPro internal simulation engine. If the FSDB file has been read in PowerPro and it has poor signal annotation, this global must be set to 0.

---

opt_vectorless_flow

opt_vectorless_flow — Enables support for vectorless flow in the design. This mode is recommended when there are no vectors to be applied on the design. Legal values are `0` and `1`. Default is `0`.

Description

The `opt_vectorless_flow` global controls support for the vectorless flow.

Note: This mode is not supported with the powerleaks flow.

Note that this global is enabled by default if the `read_saif` or the `read_fsdb` command has been specified AND both asserted primary inputs and asserted flop outputs are less than 20%.

It is recommended that this flow be enabled when the design does not contain vectors. In the vectorless flow, the results generated emphasize on potential for optimization, as shown by clock gating moves reported. The reports do not provide information about register power savings as vectors have not been applied on the design. To enable support for vectorless flow, set this global to `1`.

opt_write_compact_verilog

opt_write_compact_verilog — Controls how the generated Verilog file is written. When enabled, the generated file is compact and more readable. Legal values are `0` and `1`. Default: `0`.

Description

The `opt_write_compact_verilog` controls the quality of the Verilog file generated. Setting this global to `1` will cause constant assignments, unary assignments and binary assignments to be inlined while redundant assignments are omitted.

opt_write_rtl_allow_new_patching_scheme

opt_write_rtl_allow_new_patching_scheme — Instructs `write_rtl` to use the new style of patching RTLs. This is typically required for complex flop structures. Legal values are `0` and `1`. Default is `1`.

Description

The `opt_write_rtl_allow_new_patching_scheme` global controls how PowerPro patches the RTL for gated flops. By default, the `write_rtl` command applies a combination of patching styles, the original patching style for simple flops and the new patching style for complex flop structures. When this global is disabled, the `write_rtl` command creates a patched RTL which includes gating expressions for simple flops only. Gating expressions for complex flop structures are not written to the patched RTL.

Consider the following example.

```
always@(posedge clk)
    if (orig_en)
        flop <= d;
```

Given this is a simple flop, the `write_rtl` command will patch this using the original patching style.

```
always@(posedge clk)
   if (orig_en)
     begin
       if(pp_en)
         flop <= d;
     end
```

Now, consider another example of an RTL that contains both simple and complex flop structures.

```
always@(posedge clk)
    if (orig_en1)
        flop <= d;
....
....
 always@(posedge clk)
    if (orig_en2)
      {flop1, flop2} <= {d1, d2};
```

In this example, while there are 2 flops, it is assumed that a data gating opportunity is found only for `flop1`. As the assignment has to be broken down to include the new PowerPro enable, PowerPro cannot apply the original patching style to the complex flop. Instead, PowerPro applies the new patching style where it applies an inverted enable to retain the flop value. The patched RTL is shown below.

```
//Original patching style is applied
    always@(posedge clk)
       if (orig_en1)
         begin
           if(pp_en1)
             flop <= d;
         end
....
....
//New patching style is applied

  always@(posedge clk)
     begin
        if (orig_en2)
           {flop1, flop2} <= {d1, d2};

//PowerPro applies an inverted enable to retain the flop value
        if (!pp_en2)
           flop1 <= flop1;
     end
```

When this global is disabled, PowerPro does not patch the complex flop. In the above example, if this global was set to `0`, the patched RTL would be as follows.

```
//Original Patching Style is applied
    always@(posedge clk)
       if (orig_en1)
         begin
           if(pp_en1)
             flop <= d;
         end
....
....
//Complex flop is not patched

  always@(posedge clk)
      if (orig_en2)
        {flop1, flop2} <= {d1, d2};
```

Complex Flop Structures Written in Verilog and System Verilog

This section shows examples of complex flop structures, written in Verilog/System Verilog, where the new patching style will be applied to gate the flop.

Case 1: Flop assignments which are specified within a `for` loop.

```
always @(posedge clk or negedge reset)
      if((reset == 2'b0))
         sel_mux <= 2'b0;
      else
      begin
         for (i=0; i <= 1; i = i + 1)
              sel_mux[i] <= selin[i];
       end
```

Case 2: Flop assignments which are specified within a `while` loop.

```
integer i;

always@(posedge clk)
   begin
     i = 0;
     while(i<16)
     begin
         ff[i] <= in[i];
         i = i+1;
     end
    end
```

Case 3: Flops which are defined as tasks or functions.

```
always @(posedge clk or negedge rst_n)
if (!rst_n)
   begin
      ff1 <= 0;
      ff2 <= 0;
   end
else if (enable)
   begin
      flop(ff1, in1);
      flop(ff2, in2);
   end
```

Case 4: Flops that have a concatenation assignment or SV assignment pattern.

```
always @(posedge clk)
if(select)
   begin
      q = d;
   end
else
   {c,q} = d;
```

Case 5: Any bit -blasted flop that does not have a data gating opportunity on the complete flop. Here, it is assumed that the data gating opportunity exists only on `ff [0:7]`.

```
reg [0:15] ff;

always @(posedge clk)
     ff <= in;
```

Case 6: When flop assignment contains a complete memory but the data gating opportunity exists on part of the memory. It is assumed that the data gating opportunity exists on `ff [0]`.

```
reg [15:0] ff [0:2];

always @(posedge clk)
     ff <= in;
```

Case 7: Any composed flop that does not have a data gating opportunity on the complete flop. Here, it is assumed that the data gating opportunity exists on `obj.signal2` only.

```
typedef struct packed
{
    logic[31:0] signal1;
    logic[31:0] signal2;
} mystruct;

mystruct obj;

always @(posedge clk)
    obj <= in;
```

Case 8: When the flop index is an input.

```
input  msp;

genvar  y;
generate
      for (y = 0; y<2 ; y=y+1) begin : YLOOP
          always@(posedge clk or negedge rst)
              if(!rst)
                  pff_6[msp][y] <= 0;
              else
                  pff_6[msp][y] <= in1;
      end
endgenerate
```

Case 9: When an integer is used as the flop index without any `for/generate` loop.

```
int p;
assign p = 1;

always@(posedge clk or negedge rst)
    if(!rst)
        ff[p] <= 0;
    else
        ff[p] <= in1[p];

assign out = selff ? ff[1]: in2[1];
```

Examples of Complex Flop Structures Written in VHDL

This section shows examples of complex flop structures, written in VHDL, where the new patching style will be applied to gate the flop.

Case 1: Flop assignments which are specified within a `generate` statement.

```
gen : for i in 0 to  WIDTH  - 1 generate
   process (clk, reset)
      begin
         if ((NOT(reset)) = '1') then
            ff(i) <= '0';
         elseif (clk'EVENT AND clk = '1') then
            if (en = '1') then
               ff(i) <= in(i);
            end if;
         end if;
    end process;
end generate;
```

Case 2: Any Composed/Bit -blasted/Array -blasted flop that is not a VHDL output port. For VHDL output ports, the default patching scheme is used.

```
signal d1_ff    : std_logic_vector(2 downto 0);
signal which_ff : std_logic;
begin
  process (clk) begin
    if (rst = '1') then
      d1_ff <= "000";
      which_ff <= '0';
    elsif (rising_edge(clk)) then
      d1_ff(0 to 0) <= in1(0 to 0);
      d1_ff(1 to 1) <= in1(1 to 1);
      which_ff <= which;
    end if;
  end process;

q <= d1_ff when which_ff = '0' else in2;
```

Case 3: Flops which are defined as `procedure`.

```
architecture ff_arch of ff is
    procedure flop (signal A : in bit; signal B : out bit) is
begin
   B <= A;
end procedure flop;
begin
    process (clk) begin
       if (clk'event and clk = '1') then
           flop(d,q);
       end if;
    end process;
end architecture;
```

Case 4: Flop assignments which are specified within a `for` loop.

```
process (clk) begin
    if (rst = '1') then
      d1_ff <= "000";
    elsif (rising_edge(clk)) then
        for i in 0 to 1 loop
            d1_ff(1 downto i) <= d1(1 downto i);
        end loop;
     end if;
  end process;
```

pa_cg_integrated_flow

pa_cg_integrated_flow — Determines whether or not power numbers will be reported in the PowerPro CG/MG flow. Legal values are `0` and `1`. Default is `1`.

Description

The `pa_cg_integrated_flow` global enables reporting of power numbers in the PowerPro CG/MG flow.

Note that the `report_power` command is not supported when this global is disabled. When this global is enabled, the `report_power` command can only be called in the flow after `prototype_design` while commands that apply power analysis constraints must be called before `prototype_design`.

Some commands that apply power analysis constraints are `set_target_technology`, `set_binding`, `set_load`, `set_operating_condition`, `set_wire_load_mode` and `set_wire_load_model`. For a complete list of commands that can be used to apply power analysis constraints, refer to the *PowerPro User Manual*.

pa_clock_network_include_cgics

pa_clock_network_include_cgics — Determines whether or not CGICs will be included in the clock tree network during clock tree synthesis. By default, CGICs are included in the clock tree network. If this variable is set to exclude CGICs from the clock tree network, a separate clock tree is created using the output of the CGIC to account for the clock nets coming out of the CGICs. Possible values are `0` and `1`. Default is `1`.

Description

The `pa_clock_network_include_cgics` global determines whether or not CGICs will be included in the clock tree network during clock tree synthesis. When enabled, a single clock tree is created starting from the clock root, with the registers being the leaf nodes.

pa_clock_network_include_reg_clkarcs

pa_clock_network_include_reg_clkarcs — Determines whether the clock arc power for registers must be included in the clock tree network power during power reporting. By default, the clock arc power for registers is included in the `register` category in the `report_power` command. When this global is set to `1`, this power is moved from the `register` category to the `clock_network` category. Possible values are `0` and `1`. Default is `0`.

Description

The global `pa_clock_network_include_reg_clkarcs` determines whether the clock arc power for registers must be included in the clock tree network power during power reporting.

pa_do_not_infer_clock_tree

pa_do_not_infer_clock_tree — Setting this global helps in power estimation of the pristine clock-tree as described in the RTL. Possible values are `0` and `1`. Default is `0`.

Description

The `pa_do_not_infer_clock_tree` global specifies whether or not PowerPro should build the clock tree. If this global is set to `1`, no buffering will be done by PowerPro and the user directives of `set_load` on clock nets will be honored during clock tree power estimation.

pa_enable_power_bot_flow_in_veloce

pa_enable_power_bot_flow_in_veloce — Determines whether to enable the Powerbots flow in PowerPro. When set to `1`, the Powerbots flow will be enabled. Possible values are `0` and `1`. Default value is `0`.

Description

The `pa_enable_power_bot_flow_in_veloce` global determines whether the Powerbots flow must be enabled in PowerPro. By default, the Powerbots flow is disabled.

---

pa_frequency_scaling_factor

pa_frequency_scaling_factor — Specifies the factor by which the design frequency must be scaled. The data type of the scaling factor is `float`. Default value is `1.0`.

Description

The global `pa_frequency_scaling_factor` scales the design frequency by the factor specified. If the factor of `2` is specified, all the clock frequencies would be halved and the duration of the simulation window would be doubled. This would impact the switching power and internal power. The new internal and switching power of the design would be halved. And leakage power would remain the same.

---

pa_gate_level_flow

pa_gate_level_flow — Determines whether the power analysis flow in use is for gate level netlist. Possible values are `0` and `1`. Default value is `0`.

Description

The global `pa_gate_level_flow` is required to enable the "Gate level power analysis" flow. The flow used by the `build_design` and `prototype_design` commands is different for different settings of this global. When the global is set to `1`, PowerPro will consider the design as a gate level netlist that has no behavioural constructs. Therefore, no logic optimizations would be performed and running any sequential optimization commands is not allowed.

Note: This global is under limited production support.

---

pa_max_local_bots_for_veloce

pa_max_local_bots_for_veloce — Specifies the maximum number of local parallel processes that will be launched to read the slices from Veloce. Possible values are greater than or equal to zero (`0`). Default is `0`. In addition to these parallel processes, the main process of PowerPro will also run on the host machine.

Description

This global can be used to configure PowerPro to use multiple cores on the same machine. In PowerPro, Power Analysis makes use of this feature.

---

pa_max_remote_bots_for_veloce

pa_max_remote_bots_for_veloce — Specifies the maximum number of remote parallel processes that will be launched to read the slices from Veloce. Possible values are greater than or equal to zero (`0`). Default is `0`. In addition to these parallel processes, the main process of PowerPro will also run on the host machine.

Description

This global can be used to configure PowerPro to use multiple cores on a different machine. The setup for LSF or any other workload management system can be customized. The value of the global `host_setup_configuration` must be set correctly to allow the running of jobs on remote machines. In PowerPro, Power Analysis makes use of this feature.

---

pa_report_separate_switching_power

pa_report_separate_switching_power — Changes the output of the `report_power` command to show the breakup of Internal Power into Internal Power and Switching Power. Possible values are `0` and `1`. Default value is `1`.

Description

The global `pa_report_separate_switching_power` specifies whether to display Switching Power as a separate column in the output generated by the `report_power` command. Set this global to `0` to display Internal Power and Switching Power under the Internal Power column in the output generated by the `report_power` command.

---

pa_rtl_sim_on_glpa

pa_rtl_sim_on_glpa — Determines whether the RTL simulation activity will be used for gate-level power analysis. Possible values are `0` and `1`. Default value is `0`.

Description

To generate accurate power numbers, the commands `read_name_map_file` and `read_fsdb` must be run in this order before running the `prototype_design` command in the flow.

---

pa_sdpd_based_power_computation

pa_sdpd_based_power_computation — Determines whether the event-based power computation will be used in Gate Level Power Analysis flow. Possible values are 0 and 1. Default value is 0.

Description

The global enables the event-based power computation and can be used only in the Gate Level Power Analysis flow. It enhances the accuracy of the average power computation and is a prerequisite for using the -time_based option of the `report_power` command.

The event-based power computation is possible only if the FSDB is provided to the tool.

---

pa_select_mux_architecture

pa_select_mux_architecture — Selects the architecture of the MUX during technology mapping. The possible MUX architectures are `encoded` and `decoded`. Default is `decoded`.

Description

The `pa_select_mux_architecture` global can be used to change the architecture that has been selected in MUX synthesis during technology mapping in Power Analysis.

---

pa_use_scan_flop_for_binding

pa_use_scan_flop_for_binding — Specifies whether or not a scan flop is to be used for binding. Legal values are 0 and 1. Default is 1.

Description

The `pa_use_scan_flop_for_binding` global specifies whether or not a scan flop is to be used for binding. By default, the scan flop with the least area is used for binding. To use a non-scan flop with the least area for binding, set this global to 0.

---

ppro_cache_directory_path

ppro_cache_directory_path — This global takes the path to a directory, which would be used as a cache to store the data required for subsequent runs. If the value of this global is set, PowerPro stores the data under the directory specified by this global. Possible value is a valid path to a directory. By default, PowerPro stores the data in the directory where it was started.

Description

The `ppro_cache_directory_path` global takes the path to a directory where the data required for subsequent runs can be stored. Currently, in revamped power analysis flow, the specified path is used to store the output of pseudo-synthesis for combinational logic of each design hierarchy. If the value of this global is not set, PowerPro generates the required data in the directory where it was started. This directory needs to be preserved for the Compile-Once and Re-Use flow, which is available with revamped power analysis.

---

ppro_machine_list_file

ppro_machine_list_file — Use this global only if you want to run processes on remote machines. Store the list of machine names in a file and provide that file to this global. Each line in the file must include a machine name that has password-less access with ssh from the host machine. If multiple cores of a machine are required, add multiple lines with the same machine name in the file. Blank lines are not allowed.

Description

To run a process on a remote machine, store the machine name in a file and provide that file to the global `ppro_machine_list_file`.

ppro_new_fe_enable

Deprecated w.e.f. PowerPro 10.4. The new frontend flow is now the default flow.

ppro_new_fe_enable — This global is used to enable the new RTL elaborator in PowerPro. Possible values are `0` and `1`. Default is `1`.

Description

The global `ppro_new_fe_enable` is used to enable the new RTL elaborator in PowerPro. If the global is set to `1`, the new RTL elaborator is enabled. If the global is set to `0`, the old RTL elaborator is enabled.

required_slec_version

required_slec_version — Specifies minimum SLEC version which should be used to verify optimization changes done in PowerPro.

Description

This global can be used to get the SLEC version which is compatible with this PowerPro release and should be used to verify the changes done.

sa_default_probability

sa_default_probability — Sets the default probability value for nets whose value cannot be determined or computed. The value specified through this global does not impact the probability value of clock nets in the design. Default is `0.5`.

Description

In cases where the probability value of the net cannot be determined (because it is in a high impedance state) or where the probability value of the signal has not been specified, the `sa_default_probability` global can be used to set a probability value. This value is assumed as a default that applies to all such nets. To set a specific value for a net, use the `set_sa` command.

sa_default_tcc

sa_default_tcc — Sets the default toggle count per cycle i.e. the number of transitions per clock period, for nets whose value cannot be inferred. Default is `2` for clock nets and `0.1` for all other nets.

Description

The **sa_default_tcc** sets the number of transitions per clock period for nets whose value cannot be inferred. This value is used when the user has not specified a value for the net or when PowerPro is unable to compute the value of the net while propagating switching activity data.

Note that setting this global does not impact the toggle count per cycle value for clock nets in the design; this value is applied only to those nets whose value cannot be inferred.

When both `sa_default_tcc` and `sa_default_toggle_density` have been set, the value specified by `sa_default_toggle_density` applies.

---

sa_default_toggle_density

sa_default_toggle_density — Sets the default toggle density value to be assigned to nets when toggle density cannot be computed. Default is `0.02/ns` for clock nets and `0.005/ns` for all other nets.

Description

The **sa_default_toggle_density** allows the user to set a default toggle density value to be used for nets (number of transitions per nanosecond) when toggle density has not been assigned or cannot be computed. To assign different values for nets, use the `set_sa` command. Changing the value of this global does not impact the toggle density value of clock nets in the design.

To set default toggle count per cycle value (number of transitions per clock period), use the `sa_default_tcc` global. Note that the `sa_default_toggle_density` global has higher preference than the `sa_default_tcc` global — when both globals are defined, only the value specified by `sa_default_toggle_density` is used.

---

seq_enopt_support_limit

seq_enopt_support_limit — Sets the limit on the number of supports allowed for a clock -gating expression to be considered for enable optimization. To speed up enable optimization during observability based clock gating sequential optimization, set this to `30` or less. Default is `-1`, indicating there is no limit on number of supports by default.

Description

The **seq_enopt_support_limit** global sets the limit on the number of supports allowed for a clock -gating expression to be considered for enable optimization. By default, there is no limit on number of supports. To to speed up enable optimization during `insert_observability_logic`, it is recommended that this global be set to `30` or less.

---

seq_gen_stb_max_flop_width

seq_gen_stb_max_flop_width — Determines the maximum width of flops to be considered for generation of stability moves based on self -loop gating logic during stability based sequential optimization. Default width is `8`.

Description

By default, the `insert_stability_logic` command does not consider flops whose width is greater than `8` for generating stability moves based on self -loop based gating. To increase the scope of flops considered, set the width using the `seq_gen_stb_max_flop_width` global. A higher value will allow more flops to be gated.

---

seq_gen_stb_stability_factor

seq_gen_stb_stability_factor — Determines the stability threshold of flops to be considered for generation of stability moves based on self -loop gating logic during stability based sequential optimization. Default threshold is `.99`.

Description

The `insert_stability_logic` command generates self -loop gating logic based on the stability threshold of flops. By default, flops with a stability threshold factor greater than `.99` are considered. To change the list of flops being considered, change the stability threshold value. A lower value will allow more flops to be gated.

---

seq_generate_stable_condition

seq_generate_stable_condition — During stability based clock gating sequential optimization, this global determines whether or not comparator based stability conditions are generated for signals and flops which are highly stable. Legal values are `0` and `1`. Default is `0`.

Description

This global determines whether or not comparator based stability conditions are generated for highly stable flops and signals during `insert_stability_logic`.

In the high -effort mode of `insert_stability_logic`, if a SAIF file has been read, then this global is automatically enabled. Set this global to `0` to explicitly to turn off the generation of stable conditions in the high -effort mode. Set this global to `1` to enable generation of stability conditions in the medium and low effort modes of `insert_stability_logic`.

---

seq_obs_discard_diff_resets

seq_obs_discard_diff_resets — By setting this global to `1`, PowerPro does not add observability gating logic components that require async -override logic. Legal values are `0` and `1`. Default: `0`.

Description

By setting this global to `1`, PowerPro does not add observability gating logic components that require async -override logic, with the caveat that there will be a loss in QoR when the design includes components of observability logic that require the generation of async -override logic.

---

seq_obs_iterative_x2obs

seq_obs_iterative_x2obs — Determines whether or not enhanced observability based sequential optimization is performed in the high effort mode. Legal values are `0` and `1`. Default is `0`.

Description

When this global is enabled, additional computations are performed during observability based sequential analysis (performed only in the high effort mode), resulting in highly efficient clock gating expressions. However, as this involves complex computations, there is a possibility that this will result in higher run time during `insert_observability_logic`.

---

seq_opt_enable_cgobs_slicing

seq_opt_enable_cgobs_slicing — Enables slicing of flops in the `insert_observability_logic` command. This lets PowerPro compute distinct observability conditions for each slice of a flop, when available. Possible values are `0` and `1`. Default is `0`.

Description

When this global is set to `1`, the flops that have distinct observability conditions on separate slices are divided and gated accordingly.

---

seq_stb_self_loop

seq_stb_self_loop — Controls the generation of stability moves based on self -loop gating logic during `insert_stability_logic`. Legal values are `0` and `1`. Default is `0`.

Description

By default, the `insert_stability_logic` command does not generate stability moves based on self -loop gating logic for a flop. To enable generation of stability moves based on self -loop gating logic, set the `seq_stb_self_loop` global to `1`.

---

set_zero_replicate_to_null

set_zero_replicate_to_null — Determines how PowerPro interprets multi-concatenated expressions with non-positive repeat values. When this global is enabled, such expressions are set to NULL. Legal settings are `0` and `1`. Default: `1`.

Description

When this global is set to `1`, multi-concatenated expressions with non-positive repeat values are replaced with `NULL`. When this global is set to `0`, such expressions are replaced with `1'b0`. Consider the following example.

```
assign c = { {0{1'b1}}, a[31:0] };
assign c = { {1'bx{1'b1}}, a[31:0] };
assign c = { {1'bz{1'b1}}, a[31:0] };
assign c = { { -1{1'b1}}, a[31:0] };
```

For the above mentioned example, when this global is set to `1`, the expressions will be treated as:

```
   assign c =  { a[31:0] };
```

When set to `0`, the expressions will be treated as:

```
   assign c = { 1'b0, a [31:0] }
```

show_wildcard_expansion

show_wildcard_expansion — Shows wildcard expansions in commands. Legal settings are `0` or `1`. Default setting is `0`.

Description

When appropriate, wildcards (`*`) are automatically expanded and substitutions are individually executed.

slec_host_setup_configuration

slec_host_setup_configuration — Configures the `host_setup_configuration` global to parallelize the verification runs in SLECPro. Possible values include a valid LSF command.

Description

To run a process on the remote machine, the network-specific settings must be provided by the various options of the LSF command. For more information, refer to the description of the `host_setup_configuration` global.

slec_max_local_async_workers

slec_max_local_async_workers — Specifies the maximum number of parallel processes that will run on the local (master) machine for SLECPro. Possible values are greater than or equal to `zero` (`0`). Default is `1`.

Description

Specifies the maximum number of parallel processes that will run on the local (master) machine for SLECPro.

slec_max_remote_async_workers

slec_max_remote_async_workers — Specifies the maximum number of parallel processes that will run on the remote machine for SLECPro. Possible values are greater than or equal to `zero` (`0`). Default is `1`.

Description

Specifies the maximum number of parallel processes that will run on the remote machine for SLECPro.

---

slec_num_parallel_solver_jobs

slec_num_parallel_solver_jobs — Enables the solver-level parallelization to verify the PowerPro observability-based optimizations for memories and flops in SLECPro. Default is `0`. To enable parallelization, the value should be greater than `1`.

Description

Enables the solver-level parallelization to verify the PowerPro observability-based optimizations for memories and flops in SLECPro.

---

slecpro_new_fe_enable

slecpro_new_fe_enable — This global is used to enable the new RTL elaborator in SlecPro. Possible values are `0` and `1`. Default is `1`.

Description

The global `slecpro_new_fe_enable` is used to enable the new RTL elaborator in SlecPro. If the global is set to `1`, the new RTL elaborator is enabled. If the global is set to `0`, the old RTL elaborator is enabled.

slicing_aware_wen_synthesis

slicing_aware_wen_synthesis — Enables slicing of flops for the purpose of write-enable synthesis. This lets PowerPro compute distinct write-enable conditions for different slices of the flop, when the same write-enable condition is not available for all slices. Possible values are `0` and `1`. Default is `0`.

Note: When set to `1`, it is not recommended to use the output of the `write_rtl` command.

Description

When the global is set to `1`,

- The word-level flops, whose different bits have different write-enable conditions, are sliced so that all the bits within a slice have the same write-enable condition.
- The output of the `write_rtl` command can be syntactically incorrect or can lead to spurious falsifications.

upf_file_name

upf_file_name — This global takes the path to the UPF file that must be read. Possible value is a valid path to the UPF file.

Description

The `upf_file_name` global takes the path to the UPF file. The UPF file contains the power intent of the design. If the value of this global is not set, the PowerPro flow will continue without considering the power intent of the design. Note that only one UPF file can be read at a time.

---

upf_version

upf_version — This global is used to set the version of the UPF file. Possible values are `1.0`, `2.0`, `2.1`, and `2.2`. Default is `2.0`.

Description

The `upf_version` global is used to set the version of the UPF file. If the global is set to a specific version, the UPF file is read and analyzed according to the version. For instance, if the `upf_version` is `2.1`, the electrical or functional equivalence of supply nets or supply sets can be verified. However, if the `upf_version` is `2.0`, this capability is not available.

---

x_toggle_derate_factor

x_toggle_derate_factor — Global to control the number of toggles when the value of signal changes from '0/1' to 'x' to '0/1'. Legal values are between 0.0 and 1.0. Default value for this global is 1.0 in RTL PA flow and 0.5 in Gate Level PA flow. The value for this global can only be changed in Gate Level PA flow.

Description

Any change in value is considered a toggle by default. Thus the transition 0 -> 1 -> 0, is considered as 2 toggles, while a change in value from 0 -> X -> 0, is also considered as 2 toggles. This global can be used to control the toggle count for the case when value of the signal changes from '0/1' to 'x' to '0/1'. By default, transition from '0/1' to 'x' to '0/1' is counted as 2 toggle in RTL PA flow, while, this transition is counted as 1 toggle in Gate Level PA flow.

Chapter 4. Command Line Interface

PowerPro is run from the command line. It provides a command shell that interprets Tcl commands to read a design, optimize it, and output various reports. A Tcl script can be specified from the command line, or the Tcl commands can be entered in an interactive Tcl shell.

PowerPro can be run from the command line as follows:

```
$ powerpro –help

  Usage:  powerpro <options> <tcl-script> <script-arguments>

    -cg
          Enables CG features and supports the RTL writing capability.

    -mg
          Enables MG features and supports the RTL writing capability.

    -analyzer
          Enables Analyzer features.

    -pa
          Enables Analyzer features using revamped power analysis.

    -designer
          Enables Analyzer, CG, MG, and Exploration features.

    -optimizer
          Enables Analyzer, CG, MG, and Exploration features, and
          supports the RTL writing capability.

    -eval <string>
          Evaluates <string> in Tcl at startup.

    -help
          Shows this message and exits.

    -interactive
          Stays in the interactive shell even if <tcl-script> calls
          exit.

    -logfile <filename>
          Writes the session log to <filename> instead of the default
          powerpro.log file.

    -nocopyright
          Suppresses the verbose copyright preamble message at startup.

    -queuelic
          Waits indefinitely for the license instead of erroring out.

    -queuelic_limit <value>
          Waits for the license with a time limit, such as 30s or 1h.

    -mgls_license_file <portnumber@servername>
          Specifies the location of the MGLS license server. For
          example, 33104@server.
```

```
     -version
          Shows the tool version and exits.

    -workdir <directory>
          Writes the tool outputs to <directory> instead of the default
         ./calypto directory.

  Option flags may be abbreviated (-v for -version, -work for -workdir,
and so on). In Tcl, $argv refers to the <tcl-script> and any trailing
<script-arguments>.
```

The **powerpro** command with no parameters provides an interactive command shell. Otherwise, PowerPro runs the Tcl script, runs the command string, or issues a help message. Upon completing the script, PowerPro normally exits. The `-interactive` option forces PowerPro to remain in an interactive Tcl shell where additional commands may be executed.

PowerPro returns an error code of `0` for normal execution or non -zero for any other condition.

Using the command string option and the interactive option are useful when creating or debugging a Tcl script. For example:

```
powerpro -eval "build_design rtl.v"
```

The above command will read the modules in the `rtl.v` verilog file into PowerPro. Any errors reading the design will be written to standard out and reported in the log file. An example using the interactive option:

```
powerpro -interactive test.tcl
```

The above command starts PowerPro, executes the commands in the `test.tcl` script and remains in the interactive shell even if the script calls exit. Reporting and status commands can then be entered to explore the setup and verification results.

Appendix A. Pattern Matching Syntax

A.1. Introduction

Pattern matching is used to look up named identifiers in the design database and for specifying file names and paths. Both regex and glob styles are supported.

The regex style is the Perl -compatible regular expression style, which allows control over literals, character classes and sets, wildcards, repeats, grouping, alternatives, anchors, and escape sequences.

The glob style is the UNIX -style wildcard pattern, a simpler form of expression which allows only wildcards and character ranges.

Portions of pattern matching functionality are derived from the boost.org regular expression library; portions of the documentation within this appendix are derived from the Boost.regex documentation, copyright John Maddock 1998 -2003.

A.2. Regular Expression Syntax

The support set is as follows:

A.2.1. Literals

All characters are literals except: ., |, *, ?, +, (,), {, }, [,], ^, $ and \. These characters are literals when preceded by a \.

A.2.2. Glob

The dot character . matches any single character except newline character.

A.2.3. Repeats

A repeat is an expression that is repeated an arbitrary number of times. An expression followed by * can be repeated any number of times including zero. An expression followed by + can be repeated any number of times, but at least once. An expression followed by ? may be repeated zero or one times only. When it is necessary to specify the minimum and maximum number of repeats explicitly, the bounds operator "{}" may be used, thus "a{2}" is the letter a repeated exactly twice, "a{2,4}" represents the letter a repeated between 2 and 4 times, and "a{2,}" represents the letter a repeated at least twice with no upper limit. Note that there must be no whitespace inside the {}, and there is no upper limit on the values of the lower and upper bounds. All repeat expressions refer to the shortest possible previous subexpression: a single character; a character set, or a subexpression grouped with "()" for example.

A.2.3.1. Examples

"ba*" will match all of "b", "ba", "baaa" etc.

"ba+" will match "ba" or "baaaa" for example but not "b".

"ba?" will match "b" or "ba".

"ba{2,4}" will match "baa", "baaa" and "baaaa".

A.2.4. Nongreedy Repeats

Nongreedy repeats are possible by appending a '?' after the repeat; a nongreedy repeat is one which will match the shortest possible string.

A.2.5. Parenthesis

Parentheses serve two purposes, to group items together into a subexpression, and to mark what generated the match. For example the expression "(ab)*" would match all of the string "ababab". It is permissible for subexpressions to match null strings. If a subexpression takes no part in a match (for example if it is part of an alternative that is not taken) then both of the iterators that are returned for that subexpression point to the end of the input string, and the matched parameter for that subexpression is false. Subexpressions are indexed from left to right starting from 1, subexpression 0 is the whole expression.

A.2.6. NonMarking Parenthesis

Sometimes you need to group subexpressions with parenthesis, but don't want the parenthesis to spit out another marked subexpression, in this case a nonmarking parenthesis (?:expression) can be used. For example the following expression creates no subexpressions:

"(?:abc)*"

A.2.7. Forward Lookahead Asserts

There are two forms of these; one for positive forward lookahead asserts, and one for negative lookahead asserts:

"(?=abc)" matches zero characters only if they are followed by the expression "abc".

"(?!abc)" matches zero characters only if they are not followed by the expression "abc".

A.2.8. Alternatives

Alternatives occur when the expression can match either one subexpression or another, each alternative is separated by a |. Each alternative is the largest possible previous subexpression; this is the opposite behavior from repetition operators.

A.2.8.1. Examples

"a(b|c)" could match "ab" or "ac".

"abc|def" could match "abc" or "def".

A.2.9. Sets

A set is a set of characters that can match any single character that is a member of the set. Sets are delimited by [and] and can contain literals, character ranges, character classes, collating elements and equivalence classes. Set declarations that start with ^ contain the compliment of the elements that follow.

A.2.9.1. Examples

Character literals:

"[abc]" will match either of a, b, or c.

"[^abc] will match any character other than a, b, or c.

Character ranges:

"[az]" will match any character in the range a to z.

"[^AZ]" will match any character other than those in the range A to Z.

Note that character ranges are highly locale dependent: they match any character that collates between the endpoints of the range, ranges will only behave according to ASCII rules when the default "C" locale is in effect.

Character classes are denoted using the syntax "[:classname:]" within a set declaration, for example "[[:space:]]" is the set of all whitespace characters. The available character classes are:

alnum	Any alpha numeric character.
alpha	Any alphabetical character a -z and A -Z. Other characters may also be included depending upon the locale.
blank	Any blank character, either a space or a tab.
cntrl	Any control character.
digit	Any digit 0 -9.
graph	Any graphical character.
lower	Any lower case character a -z. Other characters may also be included

	depending upon the locale.
print	Any printable character.
punct	Any punctuation character.
space	Any whitespace character.
upper	Any upper case character A -Z. Other characters may also be included depending upon the locale.
xdigit	Any hexadecimal digit character, 0 -9, a -f and A -F.
word	Any word character all alphanumeric characters plus the underscore.
unicode	Any character whose code is greater than 255, this applies to the wide character traits classes only.

There are some shortcuts that can be used in place of the character classes:

\w in place of [:word:]

\s in place of [:space:]

\d in place of [:digit:]

\l in place of [:lower:]

\u in place of [:upper:]

Collating elements take the general form [.tagname.] inside a set declaration, where tagname is either a single character, or a name of a collating element, for example [[.a.]] is equivalent to [a], and [[.comma.]] is equivalent to [,]. The library supports all the standard POSIX collating element names, and in addition the following digraphs: "ae", "ch", "ll", "ss", "nj", "dz", "lj", each in lower, upper and title case variations. Multicharacter collating elements can result in the set matching more than one character, for example [[.ae.]] would match two characters, but note that [^[.ae.]] would only match one character.

A.2.10. Line Anchors

An anchor is something that matches the null string at the start or end of a line: ^ matches the null string at the start of a line, $ matches the null string at the end of a line.

A.2.11. Back References

A back reference is a reference to a previous subexpression that has already been matched, the reference is to what the subexpression matched, not to the expression itself. A back reference consists of the escape character \ followed by a digit 1 to 9, "\1" refers to the first subexpression, "\2" to the second etc. For example the expression "(.*)\1" matches any string that is repeated about its midpoint for example "abcabc" or "xyzxyz". A back reference to a subexpression that did not participate in any match, matches the null string.

A.2.12. Characters by code

This is an extension to the algorithm that is not available in other libraries, it consists of the escape character followed by the digit 0 followed by the octal character code. For example "\023" represents the character whose octal code is 23. Where ambiguity could occur use parentheses to break the expression up: "\0103" represents the character whose code is 103, "(\010)3 represents the character 10 followed by 3. To match characters by their hexadecimal code, use \x followed by a string of hexadecimal digits, optionally enclosed inside {}, for example \xf0 or \x{aff}, notice the latter example is a Unicode character.

A.2.13. Word operators

"\w" matches any single character that is a member of the "word" character class, this is identical to the expression "[[:word:]]".

"\W" matches any single character that is not a member of the "word" character class, this is identical to the expression "[^[:word:]]".

"\<" matches the null string at the start of a word.

"\>" matches the null string at the end of the word.

"\b" matches the null string at either the start or the end of a word.

"\B" matches a null string within a word.

A.2.14. Buffer operators

"\`" matches the start of a buffer.

"\A" matches the start of the buffer.

"\'" matches the end of a buffer.

"\z" matches the end of a buffer.

"\Z" matches the end of a buffer, or possibly one or more new line characters followed by the end of the buffer.

A buffer is considered to consist of the whole sequence passed to the matching algorithms.

A.2.15. Escape operator

The escape character \ has several meanings. Inside a set declaration the escape character is a normal character. The escape operator may introduce an operator for example: back references, or a word operator. The escape operator may make the following character normal, for example "*" represents a literal * rather than the repeat operator.

A.2.16. Single character escape sequences

The following escape sequences are aliases for single characters:

Escape sequence	**Character code**	**Meaning**
\a	0x07	Bell character.
\f	0x0C	Form feed.
\n	0x0A	Newline character.
\r	0x0D	Carriage return.
\t	0x09	Tab character.
\v	0x0B	Vertical tab.
\e	0x1B	ASCII Escape character.
\0dd	0dd	An octal character code, where dd is one or more octal digits.
\xXX	0xXX	A hexadecimal character code, where XX is one or more hexadecimal digits.
\x{XX}	0xXX	A hexadecimal character code, where XX is one or more hexadecimal digits, optionally a unicode character.
\cZ	z -@	An ASCII escape sequence controlZ, where Z is any ASCII character greater than or equal to the character code for '@'.

A.2.17. Miscellaneous escape sequences

The following are provided mostly for perl compatibility, but note that there are some differences in the meanings of \l \L \u and \U:

\w	Equivalent to [[:word:]].
\W	Equivalent to [^[:word:]].
\s	Equivalent to [[:space:]].
\S	Equivalent to [^[:space:]].
\d	Equivalent to [[:digit:]].
\D	Equivalent to [^[:digit:]].

\l	Equivalent to [[:lower:]].
\L	Equivalent to [^[:lower:]].
\u	Equivalent to [[:upper:]].
\U	Equivalent to [^[:upper:]].
\C	Any single character, equivalent to '.'.
\X	Match any Unicode combining character sequence, for example "a\x 0301" (a letter a with an acute).
\Q	The begin quote operator, everything that follows is treated as a literal character until a \E end quote operator is found.
\E	The end quote operator, terminates a sequence begun with \Q.

A.3. Wildcard syntax

The "wildcard" style is much simpler but less powerful than the "regex" style. It offers the end user a simpler way to express common match patterns. It permits the following syntax.

A.3.1. Asterisk (*)

The wildcard * character is equivalent to regex ".*", matching zero or more nongreedy occurrences of any literal.

A.3.2. Any character (?)

The wildcard ? character is equivalent to regex ., matching precisely one literal.

A.3.3. Double Asterisks (**)

When searching across design instances, the double asterisks `**` operator is made available with the `glob` style.

The intent of `**` is to match zero or more layers of hierarchy. To match one or more layers of hierarchy, use `*.**`. A `**` is only valid at the beginning of an expression or immediately after a '.'. A '.' must immediately follow `**`. A search expression may not end with `**` A search expression may contain multiple `**` and `*` operators.

For example, consider the following design instance hierarchy:

```
instance i
  ports p pp
  instance i
    ports p pp
  instance ii
    ports p pp
instance ii
```

```
ports p pp
instance i
  ports p pp
instance ii
  ports p pp
```

The following table shows some search strings and resulting matches:

**.p	i.p i.i.p i.ii.p ii.p ii.i.p ii.ii.p
ii.**.p	ii.p ii.i.p ii.ii.p
i.**.p	i.p i.i.p i.ii.p
ii*.**.p	ii.p ii.i.p ii.ii.p
.i..p	i.p i.i.p i.ii.p ii.i.p
i.**.p*	i.p i.i.p i.ii.p i.pp i.i.pp i.ii.pp
***	error (perhaps **.* was meant)
i.**	error (perhaps i.**.* was meant)
i.i**p	error (perhaps i.i*.**.*p was meant)

A.3.4. Backreference and implicit subexpressions

The wildcard style does not provide a syntax for explicitly grouping matched subexpressions for backreferencing, the wildcard `?`, `*`, and `**` characters shall be interpreted as implicit subexpression groups.

A.4. Substitution syntax

Substitution makes sense only in the presence of an accompanying pattern match. Each matching subexpression extracted from a pattern match is used towards the generation of a new string from a formatted string. Expansions are position based, ordered from left to right, counting from 1 up.

%N	Expands to the text that matched N'th sub-expression
%0	Expands to all of the current match
%&	Expands to all of the current match

Appendix B. Status of Commands and Globals

This appendix captures the status of the different commands and globals.

PowerPro Commands

Command	Added in...	Status
all_clocks	6.1	Active
all_inputs	6.1	Active
all_outputs	6.1	Active
all_registers	6.1	Active
analyze_mem_sim_traces	4.0	Deprecated in 10.4
analyze_redundant_reset	10.1	Active
bind_to_tech_cell	6.0	Active
build_db	1.0	Active
build_design	1.0	Active
config_clock_tree	6.0	Active
config_design_bin	10.2	Active
config_guidance_reports	10.2	Active
config_metrics	10.4	Active
config_multicore_setup	10.5	Active
config_opt_expr	10.4	Active
config_reset_spec	3.0	Active
config_write_rtl	2.0	Active
create_black_box	1.0	Active

Command	**Added in…**	**Status**
create_clock	1.0	Active
create_generated_clock	6.1	Active
create_mode	4.0	Active
create_mode_constraint	4.0	Active
define_memory_operation	7.1	Active
disable_msg	1.0	Active
do_write_lookahead	3.0	Deprecated in 4.0
don't_use_high_fanout_signals	6.1	Active
don't_use_high_sequential_depth_signals	6.1	Deprecated * Replaced by don't_use_max_logic_depth_signals
don't_use_max_logic_depth_signals	7.0	Active
enable_msg	1.0	Active
exclude_flop_clock_gating	4.0	Active
exit	1.0	Active
find	1.0	Active
find_lib_cell	6.0-prod-2	Active
find_nodes	3.0	Active
generate_constant_condition	3.0	Deprecated in 5.1
generate_guidance	10.2	Active
get_cg_efficiency	5.1	Deprecated in 7.2
get_clock_gating_info	1.0	Active
get_clocks	6.1	Active
get_frequency	10.1	Active
get_global	1.0	Active

Command	Added in…	Status
get_load	10.0	Active
get_mem_gating_info	3.0	Active
get_nets	6.1	Active
get_ports	6.1	Active
get_power	1.0	Deprecated in 2.0
get_sa	2.0	Active
get_transition	10.0	Active
get_version	3.0	Active
insert_cg_marker_cell	3.0	Active
insert_enable_logic	1.0	Deprecated in 2.0
insert_mem_clock_gating	3.0	Deprecated in 10.4
insert_mem_observability_gating	3.0	Active
insert_mem_sleep_gating	4.0	Active
insert_mem_stability_gating	3.0	Active
insert_observability_logic	2.0	Active
insert_stability_logic	2.0	Active
insert_static_signal_gating	7.0a	Deprecated in 10.4
limit	1.0	Active
link_design	1.0	Active
load_upf	7.0	Deprecated in 10.2
mark_shift_register	6.0-prod-2	Active
mark_synchronizer	4.0	Active
opt_infer_and_set_sync_resets	10.6	Active
parse_and_mark_timing_critical_paths	6.1	Active
pget	10.2	Active

Command	Added in...	Status
preserve_lib_module	4.0	Active
prototype_design	1.0	Active
query_execute	10.0	Deprecated in 10.1
read_db	2.0	Active
read_design	1.0	Active
read_eco_db	3.0	Active
read_fsdb	5.0	Active
read_library	1.0	Active
read_name_map_file	10.1	Active
read_previous_eco_directives	6.0-prod-2	Active
read_qwave	10.1	Active
read_saif	1.0	Active
read_sdc	6.1	Active
read_spef	7.0	Active
read_vcd	3.0	Unsupported in 10.1.
read_veloce_dataset	10.2	Active
rebind_flop_macro	6.0-prod-2	Active
report_area	1.0	Deprecated in 10.4
report_clock_gated_memories	3.0	Deprecated in 10.4
report_clock_gating	10.0	Active
report_clock_tree_power	7.0	Active
report_clocks	4.1	Active
report_data_gating	6.0	Deprecated in 10.4
report_data_gating_expression	6.0	Deprecated in 10.4
report_design	6.0-prod-2	Deprecated in 10.4

Command	Added in…	Status
report_design_toggle	10.1	Active
report_efficiency	2.0	Active
report_enable_expression	4.1	Deprecated in 10.4
report_enable_logic	1.0	Deprecated in 10.4
report_energy	10.6	Active
report_flop_toggle	10.0	Active
report_format	10.2	Active
report_globals	1.0	Active
report_hierarchy	1.0	Active
report_high_toggling_signals	7.0	Deprecated in 10.4
report_html_format	7.2	Deprecated in 10.2
report_ip_metrics	10.2	Deprecated in 10.4
report_mem_enable_expression	5.1	Deprecated in 10.4
report_mem_inputs_redundant_toggle	7.0	Deprecated in 10.4
report_memory_efficiency	3.0	Active
report_memory_gating	3.0	Deprecated in 10.4
report_memory_instances	3.0	Deprecated in 10.4
report_memory_power	3.0	Deprecated in 10.4
report_memory_redundant_access	4.0	Deprecated in 10.4
report_messages	1.0	Active
report_power	1.0	Active ** Deprecated in 2.0. Added back in 6.0.*
report_qor	4.0	Active
report_query_results	10.0	Deprecated in 10.1
report_redundant_mux_activity	7.0	Deprecated in 10.4

Command	Added in...	Status
report_redundant_reset_flops	7.1	Deprecated in 10.4
report_redundant_write_info	3.0	Deprecated in 10.4
report_run_summary	10.6	Active
report_static_signal_gating	7.0a	Deprecated in 10.4
report_toggle_profile	10.6	Active
report_violating_signals	3.0	Deprecated in 10.4
restore_session	10.0	Deprecated in 10.1
set_annotated_power	6.0	Active
set_as_memory	3.0	Deprecated in 5.0
set_as_retiming_candidate	6.0-prod-2	Active
set_async_safe	3.0	Deprecated in 7.0
set_banked_memory	7.2	Active
set_binding	7.0	Active
set_bottom_up_flow	6.0	Active
set_boundary_opt	10.0	Active
set_case_analysis	6.1	Active
set_cell_as_memory	5.1	Deprecated in 10.4
set_cg_override	2.0	Active
set_cgic_cell	2.0	Active
set_clock_domain	3.0	Active
set_constant	2.0	Deprecated in 4.0
set_current_hierarchy	10.1	Active
set_default_threshold_voltage_group	6.0-prod-2	Active
set_design_scope	6.0	Active
set_dont_care	1.0	Active

Command	Added in...	Status
set_dont_optimize	1.0	Active
set_dont_use	3.0	Active
set_existing_synchronizer	3.0	Active
set_frequency	10.7	Active
set_global	1.0	Active
set_gray_box	3.0	Active
set_hierarchy	1.0	Deprecated in 4.0
set_ignore_signal	1.0	Active
set_input_delay	1.0	Active
set_library_path	7.1_1	Active
set_load	6.0	Active
set_logic_one	6.1	Active
set_logic_zero	6.1	Active
set_mem_dont_optimize	4.1	Active
set_mem_optimize	4.1	Active
set_mem_override	3.0	Active
set_mode_configuration	10.0	Deprecated in 10.1
set_mode_override	4.0	Active
set_module_as_cgic	5.1	Deprecated in 10.4
set_multi_vth_constraint	6.0-prod-2	Active
set_observable	2.0	Active
set_operating_condition	6.0	Active
set_optimize	3.0	Active
set_output_delay	1.0	Deprecated in 2.0
set_override_stability	3.0	Active

Command	Added in...	Status
set_power_spec	10.0	Deprecated in 10.1
set_powerpro_mode	4.0	Active
set_powerpro_reset	3.0	Active
set_redundancy_explorer	10.0	Deprecated in 10.1
set_reset_signal	1.0	Deprecated in 1.1
set_rtl_to_gate_name	10.1	Active
set_sa	4.0	Active
set_scaling	10.1	Active
set_scan_enable	5.1	Deprecated in 10.4
set_stable	3.0	Deprecated in 5.1
set_start_signal	2.0	Deprecated in 4.0
set_static_signal	7.0a	Deprecated in 10.4
set_target_cgic	5.1	Deprecated in 10.4
set_target_technology	7.0	Active
set_threshold_voltage_group	6.0-prod-2	Active
set_transition	10.0	Active
set_unobservable	4.0	Active
set_user_override_cell	3.0	Deprecated in 7.1
set_user_synchronizer	3.0	Active
set_verbosity	1.0	Active
set_voltage	7.0	Active
set_wireload_mode	6.0	Active
set_wireload_model	6.0	Active
show_analyzer	2.0	Active
show_designer	10.0	Deprecated in 10.1

Command	Added in...	Status
slice_flop	3.0	Deprecated in 4.0
source	1.0	Active
unroll_loop_with_limit	4.0	Removed in 10.5a
update_read_design	6.1	Active
update_rtl	3.0	Active
validate_streaming_data	10.6	Active
validate_trace_for_ppro	10.6	Active
visualize_clock_tree	10.1	Active
weaken_enable_logic	7.1	Active
write_cgic_module	2.0	Active
write_db	2.0	Active
write_phy_db	7.2	Active
write_rtl	2.0	Active
write_saif	3.0	Active
write_session	10.0	Deprecated in 10.1
write_verify_script	2.0	Active
write_vhdl_shell	3.0	Deprecated in 5.0

PowerPro Globals

Global	Added in...	Status
alert_color	1.0	Active
bind_all_to_tech_cell	6.0	Active
design_partition_count	10.2	Active
enable_mixed_language_support	5.0	Active
enable_powerleak_flow	10.1	Deprecated in 10.6

Global	Added in...	Status
enable_runtime_filter	5.1	Active
enable_sim_parallel_process	10.2	Active
error_for_missing_modules	10.0	Active
exit_on_error	2.0	Active
expand_pla_in_gating_expression	10.1	Active
host_setup_configuration	10.1	Active
ignore_libcell_binding_check	5.0	Deprecated in 10.0
ignore_memory_libcell_binding_check	10.0	Active
max_local_process_usage	10.1	Active
max_remote_process_usage	10.1	Active
maximum_iter_count	3.0	Active
maximum_recurse_count	3.0	Active
message_wrap_at	2.0	Active
opt_add_async_powerpro_reset	4.0	Deprecated in 5.1
opt_assume_default_area_for_memories_with_zero_area	4.1	Deprecated in 7.1
opt_call_implicit_write_rtl	6.0-prod-2	Active
opt_cg_min_size	2.0	Active
opt_check_dont_opt_at_cdc_boundary_first_receive_flop	10.4	Active
opt_complex_feedback_detection	7.0	Active
opt_conservative_reset_opt	10.0	Active
opt_create_case_insensitive_names	5.1	Active
opt_create_mode_override_with_case_analysis	10.1	Active
opt_disable_cgic_en2mvm_flop_memory	10.5	Active

Global	Added in…	Status
opt_disallow_stb_c_across_multipliers	6.0-prod-2	Deprecated in 7.1
opt_dontoptimize_around_memory_flops	10.4	Active
opt_eco_invalidate_all_flops_driven_by_same_enable	4.0	Active
opt_enable_data_gating	6.0	Active
opt_enable_dont_optimize_cdc_boundary_flop_error	10.4	Active
opt_enable_frequency_flow	10.7	Active
opt_enable_pragma_support	6.1	Deprecated in 7.0
opt_enable_sequential_mode_propagation	4.1	Deprecated in 7.1
opt_filter_highly_efficient_flops	10.0	Active
opt_functional_eco_flow	7.1	Active
opt_generate_maps_for_hard_operator	7.1	Deprecated in 7.1 * Also released in PowerPro 7.0_7 – active in 7.0_7
opt_generate_negedge_async_reset	5.1	Deprecated in 7.1
opt_generate_stable_maps	3.0	Deprecated in 7.1
opt_handle_combinational_loops	6.1	Deprecated in 7.1
opt_honor_parallel_case	7.0_6	Deprecated in 7.1
opt_isolate_inout_ports	5.1	Active
opt_mark_dont_use_enum_edge	7.0	Active
opt_obs_remove_const_0_moves	10.1	Active
opt_preserve_delay_on_flop	6.0-prod-2	Deprecated in 7.1
opt_propagate_static_data_across_flops	7.1	Active
opt_skip_pprst_if_sync_reset_present	10.6	Active

Global	**Added in…**	**Status**
opt_rc_integration	3.0	Deprecated in 7.1
opt_relax_hard_operator	6.1	Active
opt_relax_reset_sync_check	6.1	Deprecated in 7.1
opt_remove_dead_enable_logic	7.1	Active
opt_report_gate_count_change	6.0	Active
opt_run_aol_verification	4.0	Deprecated in 7.1
opt_second_generation_eco_flow	5.1	Deprecated in 7.0a * Use opt_functional_eco_flow instead
opt_set_cgic_cell_verification	7.0	Active
opt_share_logic_across_cghierarchies	3.0	Deprecated in 5.1
opt_skip_pprst_if_async_reset_present	5.1	Active
opt_use_sim_for_clock_gating	10.0	Active
opt_vectorless_flow	6.1	Active
opt_write_rtl_allow_new_patching_scheme	4.0	Active
pa_cg_integrated_flow	7.0	Active
pa_clock_network_include_cgics	7.0	Active
pa_clock_network_include_reg_clkarcs	10.1	Active
pa_convert_fsdb_to_sa	6.0	Deprecated in 7.1
pa_do_not_infer_clock_tree	10.0	Active
pa_enable_power_bot_flow_in_veloce	10.5	Active
pa_frequency_scaling_factor	10.6	Active
pa_gate_level_flow	10.0	Active
pa_max_local_bots_for_veloce	10.5	Active

Global	Added in...	Status
pa_max_remote_bots_for_veloce	10.5	Active
pa_report_separate_switching_power	10.0	Active
pa_rtl_sim_on_glpa	10.1	Active
pa_sdpd_based_power_computation	10.0	Active
pa_select_mux_architecture	10.1	Active
pa_use_scan_flop_for_binding	10.0	Active
ppro_cache_directory_path	10.5	Active
ppro_new_fe_enable	10.2	Deprecated in 10.4
ppro_machine_list_file	10.4	Active
required_slec_version	3.0	Active
sa_default_probability	4.0	Active
sa_default_tcc	4.0	Active
sa_default_toggle_density	4.0	Active
second_generation_eco_flow	5.0	Deprecated in 5.1
seq_disallow_obs_propagation_along_control_paths	3.0	Deprecated in 7.1
seq_do_async_sync_xform	3.0	Deprecated in 5.1
seq_enopt_support_limit	4.0	Active
seq_generate_stable_condition	6.0	Active
seq_obs_discard_diff_resets	5.1	Active
seq_obs_iterative_x2obs	3.0	Active
seq_obs_slice	3.0	Deprecated in 4.0
seq_obs_vcd_flow_processing	5.0	Deprecated in 7.1
seq_opt_enable_cgobs_slicing	10.1	Active
seq_stb_vcd_flow_processing	5.0	Deprecated in 7.1

Global	Added in…	Status
set_zero_replicate_to_null	5.1	Active
show_wildcard_expansion	2.0	Active
slec_host_setup_configuration	10.2	Active
slec_max_local_async_workers	10.2	Active
slec_max_remote_async_workers	10.2	Active
slec_num_parallel_solver_jobs	10.2	Active
slecpro_new_fe_enable	10.2	Active
slicing_aware_wen_synthesis	10.1	Active
stb_reuse_design_flops	5.0	Deprecated in 6.0
upf_file_name	10.2	Active
upf_version	10.2	Active
use_relative_paths	1.0	Deprecated in 2.0
verification_lib_for_tech_cells	3.0	Active
x_toggle_derate_factor	10.0	Active