compile

NAME

compile

Performs logic-level and gate-level synthesis and optimization on the current design.

SYNTAX

```
status compile
    [-no_map]
    [-map_effort medium | high]
    [-area_effort none | low | medium | high]
    [-incremental_mapping]
    [-exact_map]
    [-ungroup_all]
    [-boundary_optimization]
    [-auto_ungroup area | delay]
    [-no_design_rule
     | -only_design_rule
     | -only_hold_time]
    [-scan]
    [-top]
    [-power_effort none | low | medium | high]
    [-gate_clock]
```

ARGUMENTS

-no_map

Specifies not to map the current design into the target technology library. With this option, generic Boolean equations and generic flip-flops represent the resulting design.

-map_effort medium | high

Specifies the relative amount of CPU time spent during the mapping phase of **compile**. Valid values are **medium** and **high**. You can select one value. The default is **medium**.

-area_effort none | low | medium | high

Specifies the relative amount of CPU time spent during the area recovery phase of **compile**. Valid values are **none**, **low**, **medium**, and **high**. You can select one value. The default is the specified *map_effort*.

-incremental_mapping

Specifies to attempt only incremental improvements to the gate structure of a design. Portions of a design that are already mapped are exempt from logic-level optimization, and the resulting design should be the same (if no improvements can be made) or better in terms of its design constraints. Implementations for DesignWare operators are reselected in an incremental compile run if the swap can improve the optimization cost based on the optimization constraints on the design.

-exact_map

Specifies that the sequential elements in the final design must exactly match the descriptions specified in the HDL description. SEQGEN is a technology-independent representation of a sequential element that is inferred by HDL Compiler from HDL descriptions of sequential circuits. When you specify this option, the tool attempts to find sequential elements in the target technology library that match the behavior of only the SEQGEN element (that is, no surrounding logic is considered). In the event that an exact match for a SEQGEN is not available, a different sequential element is used. Use of the **-exact_map** option would disable sequential output inversion. For more information, see **compile_seqmap_enable_output_inversion**. For improved sequential inference, use this option in conjunction with the HDL directives **sync_set_reset**, **sync_set_reset_local**, and **sync_set_reset_all**. The new attributes or directives tell HDL Compiler how to connect nets to the SEQGEN component. Use of the attributes and directives with the **-exact_map** option ensures that the results of **compile** are predictable. Use of the **-exact_map** option does not mean the QN pin won't be used in the mapped sequential element.

-ungroup_all

Collapses all levels of hierarchy in a design, except those that have the **dont_touch** attribute set.

-boundary_optimization

Optimizes across all hierarchical boundaries in the design. This can change the function of the design so that it can operate only in its current environment. If input or output ports are complemented as a result of using this option, port names are changed according to the **port_complement_naming_style** variable.

-auto_ungroup area | delay

Specifies to automatically ungroup hierarchies in the design. A hierarchy is considered for automatic ungrouping only if it satisfies all of the following criteria:

- It does not have the **dont_touch** or **ungroup** attribute set on it.
- There are no timing constraints or exceptions set on its pins.
- Its wire load model is the same as that of its parent, or the **compile_auto_ungroup_override_wlm** variable is set to **true**.
- It has fewer child cells than the value of the **compile_auto_ungroup_area_num_cells** variable if you select **area**.

If you select **area**, all hierarchies that meet the above criteria are ungrouped. The **-auto_ungroup area** option is generally used to ungroup small hierarchies in the design to improve area recovery. It does not have a significant impact on the timing of the design.

The **-auto_ungroup delay** option employs an intelligent ungrouping strategy that attempts to improve the overall timing of the design. It chooses hierarchies that are most likely to benefit from the extra boundary optimizations that ungrouping exposes. The algorithm emphasizes hierarchies containing paths that are either critical, or likely to become critical, after subsequent optimization steps.

Delay-driven auto-ungrouping offers a less CPU-intensive alternative to using the **-ungroup_all** option for improving design timing.

-no_design_rule

Determines, along with the **-only_design_rule** and **-only_hold_time** options, whether the command fixes design-rule violations before exiting. The **-no_design_rule** option specifies for the command to exit before fixing design-rule violations, thus allowing you to check the results in a constraint report before fixing the violations. The **-no_design_rule**, **-only_design_rule**, and **-only_hold_time** options are mutually exclusive. You can use only one of the options. The default is to perform both design-rule fixing and mapping optimizations before exiting.

-only_design_rule

Determines, along with the **-no_design_rule** and **-only_hold_time** options, whether the command fixes

design-rule violations before exiting. The **-only_design_rule** option specifies for the command to perform only design-rule fixing; that is, mapping optimizations are not performed. The **-no_design_rule**, **-only_design_rule**, and **-only_hold_time** options are mutually exclusive. You can use only one option. The default is to perform both design-rule fixing and mapping optimizations before exiting.

-only_hold_time

Determines, along with the **-no_design_rule** and **-only_design_rule** options, whether the command fixes design-rule violations before exiting. The **-only_hold_time** option specifies for the command to perform only hold-time fixing, ignoring other design rules. The **set_fix_hold** command must be specified for hold-time fixing to be performed. The **-no_design_rule**, **-only_design_rule**, and **-only_hold_time** options are mutually exclusive. You can select only one option. The default is to perform both design-rule fixing and mapping optimizations before exiting.

-scan

Specifies that the command is to consider the impact of scan insertion on mission-mode constraints during optimization. This option causes the command to replace all sequential elements during optimization. Some scan-replaced sequential cells might be converted to nonscan cells later in the test synthesis process because of test design-rule violations or explicit user specifications. By accounting for the impact of internal scan insertion from the start of the design process, test-ready compile eliminates the need for future reoptimization.

-top

Fixes design-rule and top-level timing violations for a design but does not perform any mapping on the design. By default, this option fixes all design-rule violations, but it only fixes the timing violations whose paths cross top-level hierarchical boundaries. If you want this option to fix timing violations for all paths, set the **compile_top_all_paths** variable to **true**.

-power_effort none | low | medium | high

Specifies the relative amount of CPU time spent during the power optimization phase of **compile**. Valid values are **none**, **low**, **medium**, and **high**. You can select one value. The default is the specified *map_effort*. Since power optimization in **compile** is enabled by power constraints, this option is ignored if there is no power constraint on the design.

-gate_clock

Enables clock gating optimization: clock gates are automatically inserted or removed. The **-gate_clock** option cannot be used in combination with the **-only_design_rule** option. When used in combination with the **-exact_map** option, it may not be possible to honor the **-exact_map** option for those registers that are involved with clock-gating optimization.

A clock-gating cell is not modified or removed if it or its parent hierarchical cell is marked **dont_touch** with the **set_dont_touch** command.

DESCRIPTION

The **compile** command performs logic-level and gate-level synthesis and optimization on the current design. Optimization is controlled by user-specified constraints on the design. These constraints describe goals for the optimization process, such as making the smallest circuit possible or trying to make specified outputs arrive by a specified time. The optimization process trades off timing and area constraints to provide the smallest possible circuit that meets specified timing requirements. Values for the area and speed of components used in synthesizing and optimizing the design are obtained from user-specified libraries.

Constraints fall into one of two categories: design rule constraints and optimization constraints. Design rule constraints reflect technology-specific restrictions that must be met for a design to function correctly. Optimization constraints reflect less critical design goals and restrictions that are desirable but not crucial for the operation of a design.

The design rule constraints are **max_transition**, **max_fanout**, **max_capacitance**, and **min_capacitance**. All other constraints are optimization constraints. Examples include maximum delay and maximum area.

During optimization, both types of constraints are considered; however, precedence is given to meeting design rule constraints. The **min_delay** and **fix_hold** constraints are treated similarly to design rule constraints, except that they have lower priority than the maximum delay constraints. The priority given to various constraints can be modified by using the **set_cost_priority** command.

Often when a design read from an HDL description is optimized, new designs modeled from the synthetic library are generated. These designs are named according to the style specified in the **synthetic_design_naming_style** variable. For details, see the **compile_variables** man page.

When the current design is hierarchical, and there are multiple design instances of a subdesign, it is not necessary to run the **uniquify** command before running the **compile** command. The **uniquify** command can still be used on multiple design instances so they can be individually optimized. For all designs marked with the **uniquify** attribute, **compile** copies and renames them according to the **uniquify_naming_style** variable.

The **compile** command does not optimize across hierarchical boundaries unless the **boundary_optimization** attribute is set to **true**. All synthetically generated designs are created with this attribute set to **true**. If boundary optimization causes ports to be complemented, port names are changed according to the **port_complement_naming_style** variable.

To customize the way the **compile** command optimizes subdesigns, use compile directives. Commands for placing these directives include **set_flatten** and **set_structure**. Command-line options pass global directives that affect how the entire design is optimized to the **compile** command.

If the current design or any of its subdesigns is represented as a state table, the **compile** command automatically performs finite state machine optimizations on these designs. Finite state machine synthesis is also controlled with compile directives placed directly on these designs by using commands such as **set_fsm_encoding** and **set_fsm_minimize**.

If the current design or any of its subdesigns contains arithmetic operations (additions, subtractions, multiplications, and comparisons), the **compile** command automatically transforms them into datapath blocks to be implemented by a datapath generator. Datapath optimization can be controlled by using the **hlo_disable_datapath_optimization** variable.

The **compile** command reports progress in real time by displaying a report. During optimization, the report shows incremental results each time a transformation is applied to the design. The default fields of the report are ELAPSED TIME, AREA, WORST NEG SLACK, TOTAL NEG SLACK, DESIGN RULE COST, and ENDPOINT.

ELAPSED TIME

Tracks the elapsed time since the beginning of the current **compile** or **reoptimize_design**.

AREA

Shows the area of the design during the optimization.

WORST NEG SLACK

Shows the worst negative slack (max_path violation) in all path groups.

TOTAL NEG

Shows the sum of the negative slack across all endpoints in the design.

DESIGN RULE COST

Measures the distance between the actual results and the user-specified design rule constraints.

ENDPOINT

Shows the endpoint currently being worked on. When a delay violation is being fixed, the object for the ENDPOINT is a pin or a port. When a design rule violation is being fixed, the object for the ENDPOINT is a net.

You can specify other fields in addition to or instead of any of the default fields. For a list of available fields and other details about specifying the compile log format, see the **compile_log_format** variable man

page.

To display the same log format as the 1998.02 version, set **compile_log_format** = "". The fields for the 1998.02 version are TRIALS, AREA, DELTA DELAY, TOTAL NEGATIVE SLACK, and DESIGN RULE COST. The new default does not have the TRIALS and DELTA DELAY fields.

TRIALS

Tracks the number of transformations that the optimizer tries before making the current selection.

DELTA DELAY

Shows the current maximum delay cost of the design, which is the sum of the worst negative slack (max_path violation) in each path group.

The log written by **compile** shows the different phases of optimization. In addition to the Resource Allocation and Mapping phases, there are four more phases. The first of these is the Delay Optimization phase, which fixes max_path violations. In the Phase 1 Design Rule Fixing phase, design-rule violations are fixed without creating any new max_path delay violations. In the Phase 2 Design Rule Fixing phase, max_path delay constraints might be impacted while fixing design-rule violations. Finally, there is an Area Recovery phase that attempts to reduce the area of the design while keeping other constraints intact. The Area Recovery phase always does some minimal cleanup, but it performs more CPU-intensive area recovery only if max_area constraints have been set on the design. If the **-only_design_rule** option is used, the Delay Optimization phase and Area Recovery phase are skipped. If the **-no_design_rule** option is used, both Phase 1 Design Rule Fixing and Phase 2 Design Rule Fixing are skipped. If the **set_cost_priority** command with the **-delay** option has been used before using the **compile** command, Phase 2 Design Rule Fixing is skipped.

The **set_local_link_library** command sets the **local_link_library** attribute on a design. The **local_link_library** attribute contains a list of design and library files that are searched before the files are specified with the **link_library** variable whenever a link operation is performed. The **target_library** variable specifies a technology library or a list of technology libraries containing the components (usually from a specific ASIC supplier) to use during optimization. The **compile** command sets the **local_link_library** attribute of the top-level design to the value of the **target_library** variable.

If **compile** is interrupted by an interrupt signal during an interactive process, it displays the following menu:

```
Please type in one of the following options:
  1 to Write out the current state of the design
  2 to Abort optimization
  3 to Kill the process
  4 to Continue optimization
Please enter a number:
```

If you select option 1, the design is written out and the menu reappears.

If the process is running in the background, you cannot use the menu. The number of interrupt signals determines what action is performed. The sequence is as follows:

```
^C Information: Preparing to interrupt optimization... (INT-5)
^C Information: Aborting Optimization... (INT-6)
^C Information: Process terminated by interrupt. (INT-4)
```

The interrupt signal can be different from one machine to another.

EXAMPLES

The following examples apply to all modes.

The following command specifies that the current design be optimized without mapping the logic:

```
prompt> compile -no_map
```

The following command optimizes the design TEST twice; once for speed and once for area:

```
prompt> current_design TEST

prompt> set_max_delay 2.0 -to [all_outputs]

prompt> compile

prompt> set_max_area 250

prompt> compile
```

The following example specifies that the command perform restricted optimizations across hierarchical boundaries:

```
prompt> compile -boundary_optimization
```

The following example demonstrates a simple optimization strategy. The commands in the example first define the current design. Design attributes and constraints are set. Optimization phases for **compile** are defined. The **compile** command is executed with the specified options.

```
prompt> current_design TRIAL
prompt> set_wire_load_model "10x10"
prompt> set_operating_conditions "WCCOM"
prompt> create_clock -period 12 \
        -waveform {0 6} CLK
prompt> set_input_delay 2 [all_inputs]
prompt> set_output_delay 3 [all_outputs]
prompt> set_max_area 580
prompt> set_structure -boolean false
prompt> set_flatten -phase true -effort medium
prompt> compile -incremental_mapping \
        -no_design_rule
prompt> report_constraint
```

SEE ALSO

alib_analyze_libs(2)
check_design(2)
create_clock(2)
insert_dft(2)
report_compile_options(2)
reset_design(2)
set_boundary_optimization(2)
set_cost_priority(2)
set_critical_range(2)
set_dont_touch(2)
set_dont_touch_network(2)
set_dont_use(2)
set_drive(2)
set_equal(2)
set_fix_hold(2)
set_fix_multiple_port_nets(2)
set_flatten(2)
set_fsm_encoding(2)
set_fsm_encoding_style(2)
set_fsm_order(2)
set_fsm_state_vector(2)
set_input_delay(2)
set_load(2)
set_local_link_library(2)
set_logic_dc(2)
set_logic_one(2)
set_logic_zero(2)
set_max_area(2)
set_max_capacitance(2)
set_max_delay(2)
set_max_fanout(2)
set_max_transition(2)
set_min_delay(2)
set_operating_conditions(2)
set_opposite(2)
set_output_delay(2)

set_prefer(2)
set_register_type(2)
set_structure(2)
set_timing_ranges(2)
set_unconnected(2)
set_wire_load_mode(2)
set_wire_load_model(2)
set_wire_load_min_block_size(2)
set_wire_load_selection_group(2)
ungroup(2)
uniquify(2)
alib_library_analysis_path(3)
compile_auto_ungroup_area_num_cells(3)
compile_auto_ungroup_count_leaf_cells(3)
compile_auto_ungroup_override_wlm(3)
compile_log_format(3)
compile_top_all_paths(3)
ungroup_keep_original_design(3)