write_file

NAME

write_file

Writes a design netlist or schematic from memory to a file.

SYNTAX

```
int write_file
    [-format output_format]
    [-hierarchy]
    [-no_implicit]
    [-output output_file_name]
    [-scenarios scenario_list]
    [-library library_name]
    [-include_anchor_cells]
    [-exclude_references reference_names]
    [-pg]
    [design_list]
```

Data Types

```
output_format        string
output_file_name     string
scenario_list        list
library_name         string
reference_names      list
design_list          list
```

ARGUMENTS

-format *output_format*

Specifies the output format of the design. Supported output formats and their descriptions are as follows:

ddc - Synopsys internal database format (the default format)
verilog - IEEE Standard Verilog
svsim - SystemVerilog netlist wrapper
vhdl - IEEE Standard VHDL

Specifying **-format svsim** causes the tool to write out only the netlist wrapper, not the gate-level design under test (DUT). To write out the gate-level DUT use **-format verilog**. For further information see the *SystemVerilog User Guide*.

-hierarchy

Writes out all the designs in the hierarchy, starting from the designs specified in *design_list*. If *design_list* is not specified, the current design's hierarchy is written. If a design hierarchy is not completely linked, the unresolved design references are not written out.

If the output format is ddc, block abstractions are not written out unless *design_list* consists of a single design that is the top of the block abstraction's hierarchy. Alternatively, *design_list* can be left empty if the current design is the top of the block abstraction's hierarchy. The **-hierarchy** and **-output** options are required when writing block abstractions to a .ddc file.

-no_implicit

Specifies not to write (save) the synthetic design hierarchy that is implicitly created during the optimization of the design. By default, the command writes designs created in the hierarchy through the use of synthetic libraries even if you do not use the **-hierarchy** option.

-output *output_file_name*

Specifies a single file into which designs are to be written. By default, the command writes each design into a separate file named *design*.*suffix*, where *design* is the name of each design (with a full UNIX path) and *suffix* is the default suffix for the specified format. The default format suffixes and their descriptions are as follows:

```
.ddc - ddc
.v   - verilog
.sv  - svsim
.vhd - vhdl
```

The **-output** option is required if the output format is .ddc and a block abstraction hierarchy is being written.

-scenarios *scenario_list*

Lists scenarios whose constraints should be included in the output file. This option applies to ddc format only. By default, the command includes all scenarios.

-library *library_name*

Writes the Synopsys database format object to the specified library.

-include_anchor_cells

Includes anchor cells created during linking (if any) when generating Verilog output. These are, by default, filtered out in Verilog output. This option is supported only by DC Explorer.

-exclude_references *reference_names*

Specifies the reference cell names, separated by space characters, for which all cell instances must not be written. The **-exclude_references** option can only be used with the **-format verilog** option.

-pg

Includes PG nets and ports in Verilog output. If the RTL was read with the **dc_allow_rtl_pg** variable set to **true**, the data obtained from the original RTL is used. Otherwise, PG connections will be inferred from the design's UPF specification, if present. For more information, see the "Instantiating RTL PG Pins in Non-UPF Designs" section in the "Analyzing and Resolving Design Problems" chapter in the *Design Compiler User Guide*. The **-pg** option can only be used with the **-format verilog** option.

design_list

Lists the designs or design files to be written. If *design_list* is omitted, the current design is used as the design list.

Block abstractions are only written to .ddc files when exactly one design is specified, or the current design is inferred, and that design is the top design of a block abstraction hierarchy.

For other output formats, such as .v, if both block abstractions and non-block abstractions exist in the design list, only the non-block abstractions are written out. However, if only block abstractions exist in the design list, the block abstractions are written out.

Designs are not removed from memory after they have been written; to remove a design from memory, use the **remove_design** command.

DESCRIPTION

This command saves the designs from memory to disk. If a design is modified in the tool, you must use the **write_file** command to save the design.

Multicorner-Multimode Support

When writing in ddc format, constraint data for all scenarios is saved by default. You can use the **-scenarios** option to specify a list of scenarios that should be saved.

EXAMPLES

The following examples show how to save designs in ddc format.

This example shows how to write out all designs in the current hierarchy. Since no output file is specified, each design is written to a separate file using the design name as the base of the file name:

```
prompt> write_file -hierarchy
Writing ddc file 'top.ddc'
Writing ddc file 'mid.ddc'
Writing ddc file 'bot.ddc'
```

This example shows how to save all designs in the current hierarchy to a single file:

```
prompt> write_file -hierarchy -output ~bill/dc/designs.ddc
Writing ddc file '/home/bill/dc/designs.ddc'
```

This example shows how to specify a list of designs to be saved:

```
prompt> write_file {ADDER MULT16}
Writing ddc file 'ADDER.ddc'
Writing ddc file 'MULT16.ddc'
```

This example shows another use of the **-hierarchy** option. All designs in the hierarchies of designs A and B are written to a single file:

```
prompt> write_file -hierarchy -output ~bill/dc/two_hiers.ddc {A B}
Writing ddc file '/home/bill/dc/two_hiers.ddc'
```

The following example shows how to specify designs by listing the design file name instead of the design names themselves. (Use **list_designs -show_file** to see the design names associated with each file.) It writes all designs that are associated with the top.ddc file.

```
prompt> write_file top.ddc
Writing ddc file 'top.ddc'.
Writing ddc file 'mid.ddc'.
Writing ddc file 'bot.ddc'.
```

The following example shows how to resolve an ambiguous file name. It is possible to have multiple designs in memory with the same name if they are associated with different files. To resolve the ambiguity you must specify the design in the form *file_name*:*design_name*. Use the absolute path to the file to avoid possible file name conflicts. In the following example, the design name "top" is ambiguous, and the ambiguity is resolved by using the file name:

```
prompt> write_file -output new.ddc top
Error: 'top' doesn't specify a unique design
        Please use complete specification: full_file_name:design_name (UID-13)
prompt> write_file -output new.ddc /home/bill/dc/top.ddc:top
Writing ddc file 'new.ddc'
```

SEE ALSO

read_file(2)
list_designs(2)
change_names(2)
define_name_rules(2)

create_block_abstraction(2)
view_write_file_suffix(3)
dc_allow_rtl_pg(3)