elaborate

NAME

elaborate

Builds a design from the intermediate format of a Verilog module, a VHDL entity and architecture, or a VHDL configuration.

SYNTAX

```
status elaborate
    design_name
    [-library library_name | -work library_name]
    [-architecture arch_name]
    [-parameters param_list]
    [-file_parameters file_list]
    [-update]
    [-ref]
```

Data Types

```
design_name      string
library_name     string
arch_name        string
param_list       list
file_list        list
```

ARGUMENTS

design_name

Specifies the name of the design to build. The design can be a Verilog module, a VHDL entity, or a VHDL configuration.

-library *library_name*

Specifies the library name to which **work** is to be mapped. By default, **elaborate** looks in the **work** library for the design to be built. Specifying **-library** allows you to temporarily change the library to which **work** is mapped.

-work *library_name*

Specifies an alias used for **-library**.

-architecture *arch_name*

Specifies the name of the architecture. In VHDL, the default architecture is the most recently analyzed architecture.

-parameters *param_list*

Specifies a list of design parameters enclosed in quotes. Parameters within the list must be separated by commas. A specification can be based on parameter order (for example, "8,7,5") or on parameter names (for example, "N=>8,M=>6"). It is acceptable to mix ordered and named parameter specifications, as long as the ordered parameters are listed first. The full *param_list* syntax is summarized in the "DESCRIPTION" section below.

-file_parameters *file_list*

Specifies a list of files that contain parameter specifications. The specifications are appended to the parameters listed in the **-parameters** option (if present). The syntax of the parameter file is identical to the syntax between the parameter-delimiters (#) in the **-parameters** option, except that the backslashes (\\) at the end of lines and the hashes (#) must be omitted. The specified files are searched for in the search_path.

-update

Reanalyzes out-of-date intermediate files if the source can be found.

-ref

Elaborates several reference designs that contain bottom design instantiations in the elaborated module, using a hierarchical elaboration flow. When the top design is instantiated, it includes all the linkage information, including interfaces, modports, and parameters, and the logic for the lower-level designs.

DESCRIPTION

The **elaborate** command builds a design from its intermediate representation using the specified parameters.

The syntax of the parameter specifications includes a subset of VHDL syntax plus the `h constant format of Verilog. VHDL accepts all but the `h format. Verilog accepts strings, integers, and `h values. Define each parameter only once; an error message is generated if a parameter is defined more than once.

Parameters are not case-sensitive, so NUM is the same as num.

For the **-parameters** option (but not in files specified by **-file_parameters**), the parameter specification must be specially delimited. The parameter specification can be enclosed in double quotation marks ("), but this is not advised if the specification itself includes string values. Instead, use the special parameter-delimiter, which brackets the parameter specification syntax. The parameter-delimiter starts and ends with `#' (hash). For multi-line strings put backslashes (\\) as the last character in the line. This is because "#" is properly recognized as a string constant.

Spaces can be inserted in the middle of the parameter string to make your script easier to read, but you must either enclose the parameter string in double quotation marks ("), or escape the spaces using backslashes (\\).

In the following example, the same parameter is passed three times, one without intermediate spaces and two using the appropriate syntax. All three cases are equivalent.

```
prompt> elaborate TOP -param width=>32,ports=>8

prompt> elaborate TOP -param "width => 32, ports => 8"

prompt> elaborate TOP -param #width\\ =>\\ 32,\\ ports\\ =>\\ 8#
```

In the following example, the second # in the first line does not terminate the parameter:

```
prompt> elaborate MY_DES -param #words=>4,str => "#",\
        name=>"my_name",\
        matrix=>"1010101010001"&\
        "0001010101010"#
```

The formal syntax for the parameter specification is as follows:

```
parameter_list:   parameter_spec [ , parameter_spec ]*

parameter_spec:   parameter_name => value_spec
                  parameter_name = value_spec
                  value_spec

value_spec:       " string_value "
                  ' character_value '
                  integer_value
                  enum_value
                  ( value_spec [ , value_spec ]* )
                  integer_value 'h hex_value
```

When the **-ref** option is specified, the **elaborate** command scans the design and groups the bottom design instantiations into several reference designs. Each reference design contains all of the instantiations of a bottom design type.

For example, if a *top* design contains several instances of bottom designs, name *bot1* and *bot2*, the **-ref** option generates two reference designs named *top_bot1* and *top_bot2*, that contain all of the instantiations of each type of lower design, respectively. The instance parameters and SV interface specializations for SystemVerilog designs are preserved in the reference design information.

Later, the reference design can be loaded and linked, forcing the elaboration of all of the specialized information required from the *top* design. The already-elaborated bottom designs can be used in a hierarchical elaboration flow.

During elaboration, the **link** command is invoked to resolve all instances in the designs. When instances cannot be resolved, the **elaborate** command issues error messages for those instances, returning a 1 status, and continues to resolve all other instances without aborting the elaboration process. Link error messages issued during elaboration are considered as warning messages, and an explicit **link** command should be executed after reading the design.

For more information about the hierarchical elaboration flow for SystemVerilog designs with interfaces, see the *SystemVerilog User Guide*.

EXAMPLES

The following example builds the design *mult* with *N* parameter set to 8, and *M* set to 3. The **-update** switch causes all out-of-date HDL files to be automatically reanalyzed.

```
prompt> elaborate -update mult -parameters "N=8,M=3"
Information: Re-analyzing the out of date package 'USER_TYPES'. (LBR-15)
Information: Re-analyzing the out of date entity 'MULT'. (LBR-15)
Information: Re-analyzing the out of date architecture 'STRUCTURE(MULT)'. (LBR-15)
Current design is now 'MULT_N8_M3'.
1
```

The following example generates a reference design for hierarchical elaboration using the **-ref** switch. The resulting design is saved to a .ddc file.

```
prompt> elaborate -ref
Running PRESTO HDLC
Presto compilation completed successfully.
Elaborated 1 reference design. Use 'link' to elaborate the required down design.
Current design is now 'top_bottom'.
Ending elaborate -ref mode. Please link the generated designs later
1

prompt> write -f ddc -hier -o top_bottom.ddc top_bottom
Writing ddc file 'top_bottom.ddc'.
1
```

The following example uses a reference design previously generated with the **-ref** switch, links it to elaborate the bottom designs, includes specialized information required by the top module, and writes out a .ddc file:

```
prompt> analyze -f sverilog bottom.sv
Running PRESTO HDLC
Searching for ./bottom.sv
```

```
Compiling source file ./bottom.sv
Presto compilation completed successfully.
1

prompt> read_ddc top_bottom.ddc
Reading ddc file 'top_bottom.ddc'.
Loaded 1 design.
Current design is 'top_bottom'.
top_bottom

prompt> link

  Linking design 'top_bottom'
  Using the following designs and libraries:
  --------------------------------------------------------------------------
  top_bottom                 ./top_bottom.ddc

Information: Building the design 'bottom' instantiated from design 'top_bottom'
        with the parameters "N=32,M=16". (HDL-193)
Presto compilation completed successfully.
Information: Building the design 'bottom' instantiated from design 'top_bottom'
        with the parameters "M=64". (HDL-193)
Presto compilation completed successfully.
1

prompt> list_designs
bottom_M64      bottom_N32_M16  top_bottom (*)
1

prompt> write -f ddc -hier -o bottom.ddc { bottom_M64 bottom_N32_M16 }
Writing ddc file 'bottom.ddc'.
1
```

SEE ALSO

analyze(2)
define_design_lib(2)
link(2)
list_designs(2)
read_file(2)
report_design_lib(2)
write_file(2)
hdlin_auto_save_templates(3)
template_naming_style(3)
template_parameter_style(3)
template_separator_style(3)